Teoria dos Grafos e o Problema da Geração de Árvores Filogenéticas

Helder Donizete da Silva Carvalho, Marcelo Martins
hdsc.br@gmail.com, marceluscoy@gmail.com

Resumo

As árvores filogenéticas são utilizadas com o intuito de auxiliar na interpretação das relações de proximidade, grau de parentesco e possíveis ancestrais em comum em um grupo de espécies. É possível criar árvores filogenéticas a partir da comparação de sequências genéticas. Com este trabalho propomos uma solução para o problema da criação de árvores filogenéticas, utilizando o conceito de árvore geradora de custo mínimo da Teoria dos Grafos, de forma a obter uma ferramenta que facilite e agilize o estudo das relações evolutivas entre diversas espécies.

1 Introdução

Na Biologia é de grande interesse analisar as mudanças ocorridas entre os ancestrais de uma espécie para seus descendentes, e também as diferenças entre os descendentes de um mesmo ancestral. Para realizar estas análises, é utilizado o conceito de árvore filogenética. O problema da geração de árvores filogenéticas consiste em encontrar a melhor representação evolutiva entre as espécies analisadas. O desafio para gerar estas árvores são as grandes quantidades de informações e cálculos que devem ser feitos.

A computação neste contexto, ajuda a automatizar e a melhorar a confiabilidade destas análises, aumentando a velocidade desses estudos. Neste trabalho propomos resolver o problema da geração de árvores filogenéticas aplicando conceitos da Teoria dos Grafos, mais especificamente o conceito de árvores geradoras de custo mínimo. Os resultados obtidos por esta estratégia se mostraram satisfatórios, levando em conta a pouca quantidade de informação utilizada e a aplicação, sem muitas alterações, dos algoritmos para a criação de árvores geradoras de custo mínimo.

Este texto está organizado em 6 seções no total. Na segunda seção encontram-se os conceitos essenciais para o compreensão do trabalho e da solução proposta. Na terceira seção são apresentados os objetivos do trabalho. Na quarta seção é apresentada a metodologia utilizada no trabalho. Na seção seguinte são apresentados os resultados obtidos através do estudo e em seguida uma conclusão do trabalho.

2 Preliminares

Antes de iniciar este estudo é necessário observar alguns conceitos para melhor compreender o problema e a solução proposta. Nas seções seguintes apresentamos alguns conceitos de Biologia, Ciência da Computação e de Biologia Computacional, considerados essenciais para nosso trabalho.

2.1 Conceitos de Biologia

A Biologia é a ciência que estuda os seres vivos, da escala molecular até o nível populacional, assim como sua interação com o ambiente [25]. Um conceito importante dentro da Biologia é o da **hereditariedade**, que afirma que os seres vivos recebem e transmitem características através da reprodução. Estas características são definidas pelos genes, que estão contidos em uma molécula de **DNA**. O DNA (ácido desoxirribonucléico) contém informações essenciais para o desenvolvimento e funcionamento de todos os seres vivos e é composto por moléculas chamadas nucleotídeos. Cada nucleotídeo, por sua vez,

é formado por um monossacarídeo, um fosfato e uma determinada base nitrogenada. As bases nitrogenadas presentes no DNA são a adenina, citosina, guanina e timina que são representadas por A, C, G e T, respectivamente [24].

Segundo Meyer e El-Hani [20], a **evolução** é a modificação das espécies ao longo do tempo. Do ponto de vista genético, evolução pode ser definida como qualquer alteração de genes ou de um conjunto de genes [11]. Estas alterações no material genético são chamadas de **mutações**, sendo causadas por falhas de cópias durante a divisão celular, exposição à radiação e/ou químicos, ou por hipermutação somática, a qual a célula causa mutações frequentemente para produzir anticorpos imunes a novos elementos [1]. De acordo com Alberts *et al.* em [1], existem vários tipos de mutações, porém, resumidamente, podemos classificá-las em três tipos, sendo: **deletérias**, **neutras** ou **benéficas**. As mutações deletérias são aquelas cujas as modificações causam prejuízo ao organismo podendo até levá-lo a morte. Genes com esse tipo de mutação geralmente tem sua frequência reduzida na população pela seleção natural. As mutações neutras são aquelas classificadas como mudanças no material genético em que a aptidão ou condição do organismo não é influenciada. Acredita-se que grande parte das mutações não tenham efeito significativo na aptidão do organismo. Por fim, as mutações benéficas são aquelas que trazem benefícios para o organismo, como um organismo sobreviver a certas condições, as quais anteriormente não era possível. É de grande interesse analisar as mutações que ocorreram dos ancestrais para seus descendentes, e também as diferenças entre os descendentes de um mesmo ancestral. Essa relação evolutiva entre grupos de organismos é estudada, em Biologia, pela **Filogenia**. O resultado do estudo da filogenia é a história evolutiva dos **grupos taxonômicos**, que são conjuntos de táxons com características semelhantes. Os **táxons** são unidades que fazem parte de um sistema de classificação científica, qualquer unidade de um sistema de classificação dos seres vivos pode ser considerada um táxon. Os grupos taxonômicos mais conhecidos são: Reino, Filo, Classe, Ordem, Família, Gênero e Espécie.

A Ciência possui grande interesse em mensurar esta história evolutiva [3]. Para atingir este objetivo, neste trabalho utilizamos o conceitos de árvore filogenética.

2.1.1 Árvores Filogenéticas

Com o objetivo de facilitar o entendimento da filogenia, se faz necessário a utilização de uma representação gráfica. Um tipo de representação gráfica muito utilizada é a de **árvore filogenética**, que é uma interpretação das relações evolutivas entre diversas espécies que possivelmente possuem um mesmo ancestral em comum. Em uma árvore filogenética, cada espécie é representada por um nó. Os nós que possuem ramificações representam o ancestral mais recente em comum para os nós seguintes. Nós que não possuem descendentes são conhecido como folhas. As folhas da árvore são chamadas de unidades taxonômicas (ou táxons), enquanto os nós que não são folhas são chamados de unidades taxonômicas hipotéticas. Adotamos, nesse trabalho, uma representação mais simples para árvores filogenéticas, conhecida como **cladograma** [2], que demonstra as relações entre os táxons como pode ser visto na Figura 1.

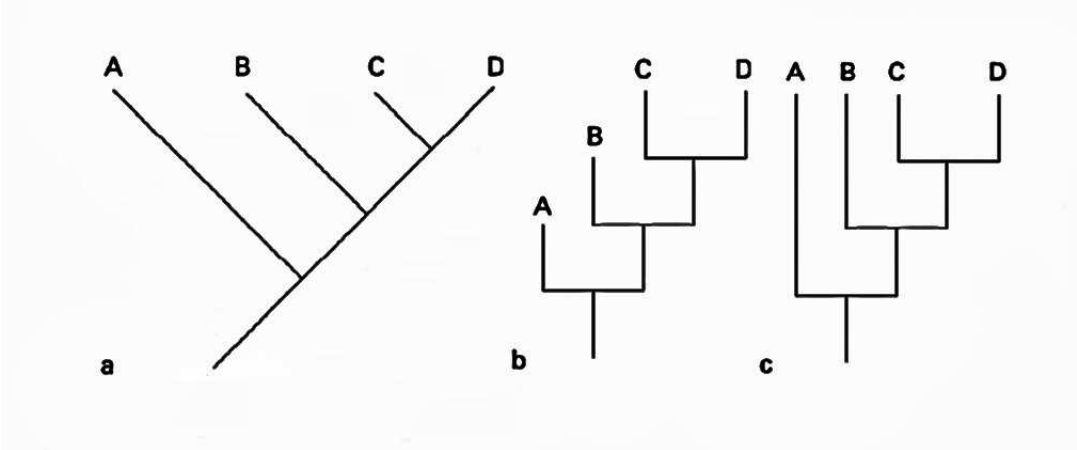

Figura 1: Exemplos de cladrogramas. Os três cladogramas representam a mesma relação entre os táxons A, B, C e D

A filogenia está fundamentada na Teoria da Evolução, que consiste em afirmar que grupos de seres vivos que apresentam similaridade de características descendem de um ancestral comum. A Teoria da Evolução foi proposta por Darwin, e tem como ideia principal que todos os seres vivos possuam um determinado grau de parentesco descendendo de um ancestral em comum [12]. Durante um tempo de aproximadamente 3,8 bilhões de anos, esse ancestral comum diferenciou-se em novas e independentes espécies [18]. Apesar dos poucos vestígios históricos, essa diferenciação deu origem à diversidade atual. Darwin começou a utilizar a representação dessa diversidade por meio de uma árvore, que vem a ser a primeira representação das relações evolutivas entre as espécies sobre a forma de árvore filogenética [13]. O problema da geração de árvores filogenéticas é considerado um dos principais problemas da Biologia Molecular [9]. Esse problema consiste em determinar as relações evolutivas de um conjunto de espécies, empregando um conjunto de dados moleculares ou morfológicos [14].

Encontrar uma árvore que representa satisfatoriamente a história evolutiva de uma espécie é um problema relevante do ponto de vista biológico. Como não se possui informações suficientes sobre as espécies já extintas, deve-se considerar cada árvore como uma hipótese. Com o aumento dos organismos de espécies diferentes estudados, o problema de obter uma árvore filogenética ideal fica ainda mais difícil já que o número de árvores a serem avaliadas consequentemente aumenta [14].

2.2 Conceitos de Ciência da Computação

A Ciência da Computação nasceu com o objetivo de buscar soluções para os problemas de diversas áreas, interagindo com diversas linhas de pensamento. Ela é aplicada em atividades de diversas áreas de pesquisa, proporcionando uma agilidade maior. Apesar de nova e em desenvolvimento, é figura presente em praticamente todas as pesquisas, tanto para novos aprendizados quanto para aperfeiçoamentos [6].

Uma das áreas muito próximas à Ciência da Computação é a Matemática. E é justamente em um ramo da Matemática, conhecido como Teoria dos Grafos, que fundamentamos a modelagem da solução proposta neste trabalho. A Teoria dos Grafos estuda objetos combinatórios conhecidos como grafos, que servem como ferramenta de modelagem para muitos problemas em várias áreas de pesquisa [15]. Segundo Bondy e Murty em [4], um **grafo** G é definido por uma dupla de $G(V, E)$, onde $V(G)$ é um conjunto de objetos denominados **vértices** e $E(G)$ é um conjunto de pares não ordenados de V, chamados de **arestas**. O número de vértices e arestas de um grafo são denominados **ordem** e **dimensão**, respectivamente, e denotados por $v(G)$ e $e(G)$. Se e é uma aresta, denotamos e por $e = uv$, onde u e v são vértices de G e são ditos extremos de e. Pode-se atribuir custos às arestas através de uma função $f(u, v) \rightarrow \mathbb{R}$ [8].

As definições e conceitos na Teoria dos Grafos são melhor compreendidos utilizando representações gráficas, sendo cada vértice representado por um ponto distinto no plano e cada aresta representada por uma linha que une um par de vértices, como representado na Figura 2. Na Figura 2, os vértices a e b, por serem os extremos da aresta ab de custo 15, são **adjacentes** entre si. Duas arestas quando incidentes a um mesmo vértice, também são ditas adjacentes, como por exemplo, as arestas ac e ad da Figura 2. Quando mais de uma aresta ligam um mesmo par de vértices, essas arestas são chamadas de **paralelas**, e quando uma aresta possui seus extremos em um mesmo vértice, é chamada de **laço**. Na Figura 2, as arestas que incidem nos vértices f e h, são exemplos de arestas pararelas, enquanto a aresta que incide no vértice b, de custo 10, é um laço. Dizemos também que um grafo é **simples** se não possuir laços nem arestas paralelas e **trivial** se possui um único vértice e nenhuma aresta [4].

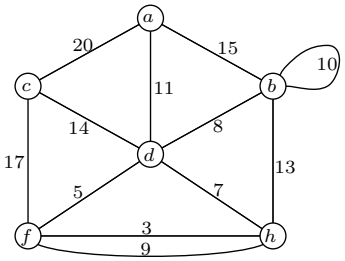

Figura 2: Grafo $G(V, E)$, com $V(G)$= $\{a, b, c, d, f, h\}$ e $E(G)$= $\{ab, ac, ad, bb, bd, bh, cd, cf, dh, df, fh, fh\}$, e os respectivos custos das arestas $E(G)$=$\{$ 15, 20, 11, 10, 8, 13, 14, 17, 7, 5, 3, 9$\}$

O **grau** de um vértice v, denotado por $d(v)$, é o número de arestas que incidem em v. Cada laço é contabilizado como duas incidências no vértice. Um grafo é dito k**-regular** se todos os seus vértices

possuem grau k. Um grafo é **completo** quando há uma aresta entre cada par de seus vértices. Seja n o número de vértices do grafo, um grafo completo será $(n-1)$-regular e denotado por K_n [4].

Um **caminho** em um grafo G é uma trajetória entre dois vértices. Para um caminho existir é necessário que tenha uma **sêquencia** $W = v_0e_0v_1e_1v_2e_2...e_{k-1}v_k$ finita e não nula, cujos os termos são vértices e arestas alternadamente, sendo o tamanho dessa sequência positivo. Quando se obtém um caminho fechado, onde o vértice de origem é o mesmo que o de término, obtem-se um **circuito** ou **ciclo**. Dizemos que um grafo $G(V, E)$ é **conexo** se ele contém pelo menos um caminho ligando cada par de seus vértices [4].

Dados dois grafos G e G', diz-se que G' é **subgrafo** de G e que G é **supergrafo** de G' quando $V(G') \subseteq V(G)$ e $E(G') \subseteq E(G)$. Quando $V(G') = V(G)$ dizemos que G' é **subgrafo gerador** de G [8]. Na Teoria do Grafos, uma **árvore** é um grafo conexo e sem ciclos, e uma **floresta** é um conjunto de árvores em um mesmo grafo. Uma **árvore geradora** de G é um subgrafo gerador de G que é uma árvore [16]. Pode-se atribuir um custo a uma árvore geradora $T = (V, E(T))$ de G, como sendo a soma dos custos das arestas $E(T)$ [22]. Uma **árvore geradora de custo mínimo** ou MST (do inglês *Minimum Spanning Tree*) é uma árvore geradora que possui o custo mínimo dentre todas as árvores geradoras de G.

Além dos conceitos da Teoria dos Grafos apresentados, também precisamos definir alguns outros conceitos chave para entender a solução proposta para o problema. Um **alfabeto** $\sum$ é definido como um conjunto finito e não-vazio de caracteres. Dado um alfabeto qualquer, uma **sequência** s é uma concatenação de caracteres desse alfabeto. O **tamanho** de uma sequência s é a quantidade de caracteres que a compõe e é denotado por $|s|$. Quando uma sequência possui seu tamanho igual a zero dizemos que é uma **sequência vazia**. Um **segmento** $s[i..j]$ de s, para $1 \leq i \leq j \leq |s|$, é uma sequência formada pelos elementos $s[i], s[i+1], s[i+2]$,..., $s[j]$. Uma **subsequência** de s é uma sequência obtida a partir de s com possíveis remoções de caracteres de s. É importante observar que, enquanto um segmento de s é uma série de caracteres consecutivos de s, uma subsequência pode ser constituída de caracteres que não necessariamente estejam consecutivos em s. Dizemos também que um segmento $x[i..j]$ é **prefixo** de uma sequência s quando x é formado pelos elementos de s, onde $i = 1$ e $j \leq |s|$ [5].

A partir de duas sequências s e t, podemos obter uma relação entre elas que chamamos de **alinhamento**. O processo de obtenção de um alinhamento entre s e t, consiste em introduzir **espaços** ou ***gaps***, representados pelo símbolo "_", dentro ou nas extremidades das sequências, deslocando alguns segmentos, de modo que ao final desse processo elas possuam o mesmo tamanho. Estes espaços podem ser inseridos em qualquer posição de s e t, de maneira que, $s[j]$ e $t[j]$, numa determinada posição j, não sejam iguais a "_". Esta propriedade é chamada de "alinhamento livre de colunas em branco" [17].

Considere as sequências *s=qacdbd* e *t=qawxb*. Um possível alinhamento entre essas duas sequências é mostrado na Figura 3.

q	a	c	_	d	b	d
q	a	w	x	_	b	_

Figura 3: Possível alinhamento entre as sequências *s=qacdbd* e *t=qawxb*

Um exemplo de alinhamento entre duas sequências é mostrada na Figura 3. Para cada posição das sequências alinhadas, pode ser atribuido um valor, de acordo com uma função de pontuação que leva em consideração se as posições sobrepostas possuem caracteres idênticos, diferentes ou espaços. A partir disso é possível atribuir um valor para o alinhamento. Mais detalhes sobre alinhamento entre duas sequências são descritos na Seção 4.1.

3 Objetivos

Neste trabalho, implementamos uma solução para o problema da geração de árvores filogenéticas utilizando os conceitos computacionais definidos na Seção 2.2. O objetivo é desenvolver uma ferramenta computacional, a partir da modelagem proposta, utilizando árvores gerados de custo mínimo, para a geração de árvores filogenéticas a partir de sequências de DNAs dos genes de algumas espécies.

Os resultados obtidos pela ferramenta podem ser utilizados como indicadores da relação entre alguns genes, que possuem a mesma função, em espécies distintas.

4 Metodologia

Para resolver o problema da geração de árvores filogenéticas utilizamos conceitos da Teoria dos Grafos, mais especificamente aplicando algoritmos para encontrar uma MST em um grafo que representam as relações entre espécies. Esses grafos são grafos completos, onde os vértices representam as espécies e as arestas desse grafo possuem um custo que simboliza a similaridade entre as espécies. Essa similaridade é calculada através do alinhamento das sequências de DNA das espécies.

Mais detalhes da solução proposta para o problema são descritos nas seções seguintes.

4.1 Alinhamento e Pontuação

A comparação entre duas sequências é uma das operações mais antigas na biologia computacional, que serve como base para outros tipos de manipulações mais complexas [26]. O alinhamento entre duas sequências, como já mencionado na Seção 2.2, consiste em inserir espaços, em uma ou ambas as sequências, de modo que ao final desse processo elas possuam o mesmo tamanho e uma possa ser disposta sobre a outra.

Uma vez que as duas sequências estejam com o mesmo tamanho é possível atribuir uma pontuação para esse alinhamento de acordo com os caracteres de suas colunas. As colunas de um alinhamento podem possuir dois caracteres iguais (*match*), dois caracteres diferentes (*mismatch*) ou ainda um caractere e um espaço (*gap*). Utilizamos uma função de pontuação $\omega(a, b) \rightarrow \mathbb{R}$, que tem a finalidade da associar um valor numérico para a relação existente entre os caracteres a e b de uma coluna do alinhamento, *match*, *mismatch* ou *gap*. A pontuação do alinhamento é dada pela soma das pontuações de todas suas colunas. Para calcular o alinhamento entre duas sequências, α e β, utilizamos o algoritmo proposto por Saul Needleman e Christian Wunsch. A ideia desse algoritmo é armazenar em uma matriz M os valores ótimos dos alinhamentos entre todos os prefixos das duas sequências envolvidas, utilizando uma função de pontuação.

O algoritmo de Needleman-Wunsch trabalha com uma matriz M com $n + 1$ colunas e $m + 1$ linhas, indexadas por $\{0 \dots n\}$ e $\{0 \dots m\}$, onde n é o tamanho da sequência α e m o tamanho da sequência β. A primeira etapa consiste em inicializar algumas posições de M. A posição $M[0, 0]$ recebe o valor 0, porque representa o alinhamento dos dois prefixos vazios de α e β. Cada posição da coluna 0 é preenchida de acordo com a regra $M[i, 0] \leftarrow gap \times i$ porque representa o alinhamento de cada prefixo de β com um prefixo vazio de α, e cada posição da linha 0 de acordo com a regra $M[0, j] \leftarrow gap \times j$, pois representa o alinhamento de cada prefixo de α com um prefixo vazio de β [21].

Considere as sequências $\alpha = \{A, C, T, G, G, G, T, C, A, A, C\}$, $\beta = \{A, T, T, G, G, C, C, A, C\}$ e o valor de $gap = -5$, um exemplo de inicialização da matriz do algoritmo de Needleman-Wunsch pode ser visto na Figura 4.

	α	A	C	T	G	G	G	T	C	A	A	C
β	0	-5	-10	-15	-20	-25	-30	-35	-40	-45	-50	-55
A	-5											
T	-10											
T	-15											
G	-20											
G	-25											
C	-30											
C	-35											
A	-40											
C	-45											

Figura 4: Inicialização da matriz de alinhamento das sequências $\alpha = \{A, C, T, G, G, G, T, C, A, A, C\}$ e $\beta = \{A, T, T, G, G, C, C, A, C\}$, com valor de $gap = -5$.

A próxima etapa do algoritmo consiste no preenchimento da matriz utilizando a Recorrência 1.

$$M[i, j] \leftarrow \max \begin{cases} M[i, j-1] + \omega(\_, \alpha[i]), \\ M[i-1, j-1] + \omega(\alpha[j], \beta[i]), \\ M[i-1, j] + \omega(\beta[j], \_) \end{cases} \quad (1)$$

Após o completo preenchimento da matriz, o valor da similaridade entre as duas sequências está armazenado na última posição da matriz, ou seja, na posição $M[m, n]$.

Considerando as sequências α e β do exemplo anterior e os valores para *match*, *mismatch* e *gap* iguais a 3, -2 e -5, respectivamente, podemos ver a matriz de alinhamento entre as duas sequências, preenchida utilizando a Recorrência 1, na Figura 5.

	α	A	C	T	G	G	G	T	C	A	A	C
β	0	-5	-10	-15	-20	-25	-30	-35	-40	-45	-50	-55
A	-5	3	-2	-7	-12	-17	-22	-27	-32	-37	-42	-47
T	-10	-2	1	1	-4	-9	-14	-19	-24	-29	-34	-39
T	-15	-7	-4	4	-1	-6	-11	-11	-16	-21	-26	-31
G	-20	-12	-9	-1	7	2	-3	-8	-13	-18	-23	-28
G	-25	-17	-14	-6	2	10	5	0	-5	-10	-15	-20
C	-30	-22	-14	-11	-3	5	8	3	3	-2	-7	-12
C	-35	-27	-19	-16	-8	0	3	6	6	1	-4	-4
A	-40	-32	-24	-21	-13	-5	-2	1	4	9	4	-1
C	-45	-37	-29	-26	-18	-10	-7	-4	4	4	7	7

Figura 5: Matriz de alinhamento entre as sequências $\alpha = \{A, C, T, G, G, G, T, C, A, A, C\}$ e $\beta = \{A, T, T, G, G, C, C, A, C\}$, com valores de $gap = -5$, $mismatch = -2$ e $match = 3$.

A Figura 6 mostra as sequências α e β sobrepostas, de acordo com a matriz de alinhamento da Figura 5.

i	=	0	1	2	3	4	5	6	7	8	9	10	
α	=	A	C	T	G	G	G	T	C	A	A	C	
β	=	A	T	T	G	G	C	-	C	-	A	C	
$\omega(\alpha[i], \beta[i])$	=	3	-2	3	3	3	-2	-5	3	-5	3	3	$= 7$

Figura 6: Resultado do alinhamento das sequências $\alpha = \{A, C, T, G, G, G, T, C, A, A, C\}$ e $\beta = \{A, T, T, G, G, C, C, A, C\}$, correspondente à matriz da Figura 5

4.2 Árvores Geradoras de Custo Mínimo

A solução proposta para a geração de árvores filogenéticas nesse trabalho tem sua base na criação de árvores geradoras de custo mínimo. Com isso, foram implementados três algoritmos para esse propósito, de tal forma que pudéssemos analisar e comparar os resultados.

O primeiro algoritmo estudado para criação de uma árvore geradora de custo mínimo foi o algoritmo de Boruvka. Este algoritmo trabalha desenvolvendo, simultaneamente, uma floresta minimal no grafo G. Inicialmente cria-se uma floresta onde cada vértice de G é uma árvore trivial em F. Enquanto F não formar uma única árvore, seleciona-se, para cada árvore T em F, uma aresta de menor custo que una duas árvores de F e que não forme um circuito [16], unindo somente conjuntos diferentes. Um pseudocódigo para o algoritmo de Boruvka é apresentado no Algoritmo 1.

Algoritmo 1: MST-Boruvka

```
1 Seja F_0 uma floresta inicial com n subárvores T_j, j = 1, ..., n de G (todos os nós isolados)
2 i ← 0
3 enquanto F_i não for uma única árvore, para cada T_j ∈ F_i faça
4 |   Determine a menor aresta (x_α, y_α) incidente em T_j
5 |   se x_α ∈ T_j e y_α ∉ T_j então
6 |   |   Faça F_{i+1} ← F_i ∪ [⋃_α (x_α, y_α)]
7 |   fimSe
8 |   i ← i + 1
9 fimEnquanto
```

Para a implementação desse algoritmo, foi utilizado o conceito de conjuntos. Conjuntos, nesse contexto, trata-se de uma coleção de elementos bem definidos, e que não possuem nenhum elemento em comum. Para melhor controle dos conjuntos, optamos por trabalhar com algumas coleções do JAVA, com isso foi necessário a utilização do tipo de dado *Iterador*, também do JAVA, pois ele permite que seja possível alterar esses conjuntos enquanto percorremos os seus elementos.

Foi desenvolvido um método que transforma cada vértice em um conjunto, onde cada conjunto representa um subárvore T, como também um método para verificar se os vértices x_α e y_α estão ou não em um mesmo conjunto. Também foi criado um método que realiza a união de dois conjuntos, unindo duas subárvores T a partir dos vértices x_α e y_α. Outro método criado realiza a busca entre as arestas que saem do conjunto que está sendo formado e elege a de menor custo para que os conjuntos relacionados a ela se unam como citado anteriormente.

O segundo algoritmo estudado foi o algoritmo de Kruskal. A ideia desse algoritmo é, a cada passo, buscar uma aresta segura para adicionar a árvore A, que está sendo desenvolvida a partir de um grafo G, encontrando entre todas as arestas as que conectam quaisquer dois subgrafos existentes em A, sendo essa aresta a de menor custo. O algoritmo inicia com o subgrafo A com $V(G)$ vértices e nenhuma aresta. Em seguida, as arestas de $E(G)$ são ordenadas pelo custo, do mais baixo ao mais alto. Após realizado os passos anteriores, para cada aresta uv de E, é verificado se u e v pertencem ao mesmo subgrafo. Se pertencerem, então a aresta não pode ser adicionada a A sem criar um ciclo, e a aresta é descartada. Caso contrário, os dois vértices pertencem a subgrafos diferentes, e a aresta pode ser adicionada a A. É apresentado um pseudocódigo para o algoritmo de Kruskal no Algoritmo 2 [10].

Algoritmo 2: MST-Kruskal

```
1  V(A) ← V(G)
2  Adicione as arestas de G em H = [h_i], i = 1, 2, ..., m
3  E(A) ← h_1
4  remove o elemento do topo do heap H
5  enquanto |H| > 0 faça
6  |  se E(A) ∪ h_i é um grafo acíclico (árvore) então
7  |  |  E(A) ← E(A) ∪ h_i
8  |  fimSe
9  |  remove o elemento do topo do heap H
10 fimEnquanto
```

Assim como feito no algoritmo de Boruvka, também foi utilizado na implementação do algoritmo de Kruskal o conceito de conjuntos. Inicialmente cada vértice de G é transformado em um conjunto. Em seguida as arestas são adicionadas em um heap de prioridade mínima H para que a seleção da aresta de menor custo seja mais efeciente. Para verificar se a aresta selecionada em H não formará um ciclo no conjunto que está sendo formado, é verificado se os vértices desta aresta estão contidos em conjuntos diferentes. Caso estejam, é realizada a união entre os conjuntos e em seguida essa aresta é removida do heap.

O último algoritmo estudado para criação de uma MST foi o algoritmo de Prim. Esse algoritmo inicia com uma árvore A vazia e escolhe um vértice arbitrário r de G para adicionar em A. Após esse passo inicial, a árvore A é expandida, vértice a vértice, até alcançar todos os vértices de G. Essa expansão é feita procurando uma aresta vw de custo mínimo, tal que $v \in V(A)$ e $w \notin V(A)$, e adicionando o vértice w à árvore A. A estratégia do algoritmo de Prim se baseia em, a cada etapa, adicionar à árvore S uma aresta que acrescenta o menor valor possível ao custo total da árvore [10]. Um pseudocódigo desse algoritmo é apresentado no Algoritmo 3.

Para seleção da menor aresta é utilizado o mesmo método mencionado no algoritmo de Boruvka. Esse método tem a finalidade de identificar a aresta de menor custo que relaciona com os conjuntos que ainda não foram adicionados ao conjunto solução para poder incluir os demais vértices que ainda não selecionados.

4.3 Geração de Árvores Filogenéticas

De modo resumido, a solução implementada nesse trabalho engloba todos os conceitos abordados nas seções anteriores. Primeiramente é realizado o alinhamento entre as sequências, duas a duas, e produzido

Algoritmo 3: MST-Prim

```
1 Escolha um vértice qualquer r ∈ V(G)
2 V(A) ← {r}
3 enquanto |V(A)| < |V(G)| faça
4 |   Encontrar a menor aresta (vw) ∈ E(G) tal que w ∈ V(A), v ∈ S
5 |   V(A) ← V(A) ∪ {v}
6 |   E(A) ← E(A) ∪ (v, w)
7 fimEnquanto
```

um grafo completo K representando as relações de similaridade entre elas, onde nesse grafo os vértices são as espécies analisadas e as arestas a similaridade. Vale ressaltar que a similaridade obtida pelo algoritmo de Needleman-Wunsch indica o quão parecidas são as duas sequências. Dessa forma, quanto maior a similaridade entre as sequências, maior vai ser o valor devolvido pelo algoritmo. Para que os algoritmos que criam as árvores geradoras de custo mínimo agrupem as sequências mais similares de forma correta, multiplicamos cada similaridade por -1 de tal forma que a maior similaridade se transforme no menor valor e a menor similaridade no maior valor. Em seguida é aplicado um dos algoritmos para a criação de árvore geradora de custo mínimo em K. A partir da árvore obtida no passo anterior é criado um cladograma, representando a árvore filogenética das sequências.

No passo da criação do cladograma utilizamos o programa R, que é um ambiente e uma linguagem para computação estatística e geração de gráficos. Ele é um *software* de código aberto e fornece uma ampla variedade de técnicas estatísticas e gráficas, oferecendo suporte para trabalhos com símbolos e fórmulas matemáticas [23]. O programa R utiliza uma matriz baseada na árvore geradora de custo mínimo para poder desenhar o cladograma, a partir de funções específicas primeiramente para calcular qual espécie ficará próxima de qual espécie e em seguida desenhar o cladograma.

Para a correta execução da ferramenta é necessário informar alguns argumentos. O primeiro argumento requerido é um valor inteiro n, com $n > 2$, que representa a quantidade de sequências que serão analisadas pela ferramenta. Os próximos n argumentos deverão ser a localização dos arquivos contendo as sequências a serem analisadas. Em seguida, deverão ser informados os valores de *gap*, *match* e *mismatch*, respectivamente. Por fim, o último argumento é o nome do algoritmo que será utilizado para criar a árvore geradora de custo mínimo, que pode ser Boruvka, Kruskal ou Prim.

5 Resultados

Após a implementação da nossa ferramenta, e, antes de realizar os testes com as amostras reais de DNA, utilizamos um conjunto de testes contendo sequências criadas artificialmente sobre o alfabeto Σ, correspondendo ao alfabeto da língua portuguesa, contendo os caracteres de A a Z. Para cada sequência s criada artificialmente no conjunto de testes, eram criadas mais cinco sequências a partir de alterações em s.

Cada sequência s criada possuia um tamanho variando entre 2000 e 7000 caracteres. Em seguida, eram criadas três novas sequências a partir de alterações dos caracteres em s. Essas alterações foram realizadas trocando um caractere de s por qualquer caractere de Σ, sem modificar o tamanho da sequência. Foram criadas sequências com 20, 50 e 75% de alterações. Depois disso, ainda eram criadas mais duas sequências a partir de operações de inserção ou remoção de caracteres em s. Eram realizadas, de forma aleatória, entre 200 e 700 operações de inserção ou de remoção. A decisão do tipo de operação a ser realizada também era tomada de forma aleatória. Nota-se que essas últimas duas sequências criadas poderiam ter tamanhos diferentes com relação às outras sequências.

Foram criados vários conjuntos como o descrito acima e colocamos nossa ferramenta para executar com essas sequências. A ideia é que, ao executar nossa ferramenta para essas sequências criadas artificialmente, as sequências criadas a partir de uma mesma sequência s fiquem próximas na árvore filogenética gerada.

A título de ilustração, destacamos um dos testes realizados com sequências artificiais pela ferramenta. Para esse teste foram usados quatro grupos de sequências como os descritos nesta seção, denominados, respectivamente, de A, B, C e D. A descrição de cada uma das sequências utilizadas nesse teste podem ser vistas na Tabela 1, onde o segundo caractere do nome de cada sequência determina em qual grupo

ela está.

Sequência	Descrição	Tamanho	Ins.	Rem.	Alt.
SAorig	Sequência original	4177	0	0	0
SA20	Alteração de 20%	4177	0	0	835
SA50	Alteração de 50%	4177	0	0	2089
SA75	Alteração de 75%	4177	0	0	3133
SAir1	Inserções e Remoções	4188	348	336	0
SAir2	Inserções e Remoções	4159	217	234	0
SBorig	Sequência original	3335	0	0	0
SB20	Alteração de 20%	3335	0	0	623
SB50	Alteração de 50%	3335	0	0	1567
SB75	Alteração de 75%	3335	0	0	2350
SBir1	Inserções e Remoções	3346	284	273	0
SBir2	Inserções e Remoções	3304	96	176	0
SCorig	Sequência original	4097	0	0	0
SC20	Alteração de 20%	4097	0	0	819
SC50	Alteração de 50%	4097	0	0	2049
SC75	Alteração de 75%	4097	0	0	3073
SCir1	Inserções e Remoções	4081	160	176	0
SCir2	Inserções e Remoções	4084	177	190	0
SDorig	Sequência original	6515	0	0	0
SD20	Alteração de 20%	6515	0	0	1303
SD50	Alteração de 50%	6515	0	0	3258
SD75	Alteração de 75%	6515	0	0	4886
SDir1	Inserções e Remoções	6539	190	176	0
SDir2	Inserções e Remoções	6531	156	139	0

Tabela 1: Descrição de um conjunto de sequências criadas artificialmente para testar a ferramenta desenvolvida. Na coluna Sequência são listados os nomes de cada sequência. Na coluna Descrição são especificadas quais alterações foram feitas para cada sequência em relação a orignal. Na coluna Tamanho é informado o tamanho de cada sequência. Nas colunas Ins., Rem. e Alt. são especificicadas as quantidades de inserções, remoções e alterações, respectivamente.

A partir das sequências descritas na Tabela 1, executamos a ferramenta para gerar as árvores filogenéticas utilizando os três algoritmos mencionados na Seção 4.2. Os três cladogramas gerados pelos três algoritmos são idênticos, o que já era de se esperar uma vez que os três algoritmos devem gerar a mesma MST. Na Figura 7 é mostrada a árvore gerada pela ferramenta.

Analisando do cladograma apresentado na Figura 7, podemos observar que as sequências geradas a partir de uma mesma sequência s foram corretamente agrupadas entre elas. Com relação às sequências do grupo A, percebemos que a sequência SAorig está mais próxima das sequências $SAir2$, $SAir1$, $SA20$, $SA50$ e $SA75$, nessa ordem. Isso era esperado, pois a sequência $SAir2$ é a que possui menos modificações em relação à sequência original e consequentemente é mais similiar. Seguindo esse raciocínio as sequências $SAir1$, $SA20$, $SA50$ e $SA70$ estão mostradas da com menos modificações para a com mais modificações, ou seja, da mais similar para a menos similar. Esse comportamento também pode ser observado entre as sequências dos grupos B e D. Já entre as sequências do grupo C houve uma troca de posições entre as sequências $SCir1$ e $SCir2$. Essa troca era esperada visto que a sequência $SCir1$ possui menos modificações que a $SCir2$, como observado na Tabela 1.

O tempo gasto pela ferramenta para gerar as árvores filogenéticas utilizando o conjunto de sequências artificiais, para os três algoritmos, foi em média dois minutos por execução. Pudemos observar que o algoritmo de Boruvka foi ligeiramente mais rápido, seguido pelo algoritmo de Prim e o de Kruskal.

Após realizados os testes com as sequências artificiais, foi definido um conjunto com amostras reais para execução da ferramenta. Esse conjunto foi criado a partir dos dados do projeto ENCODE (*Encyclopedia of DNA Elements*), que é um projeto internacional financiado pelo *National Human Genome Research Institute*, nos Estados Unidos, com o objetivo de encontrar todos os elementos funcionais do genoma humano [27].

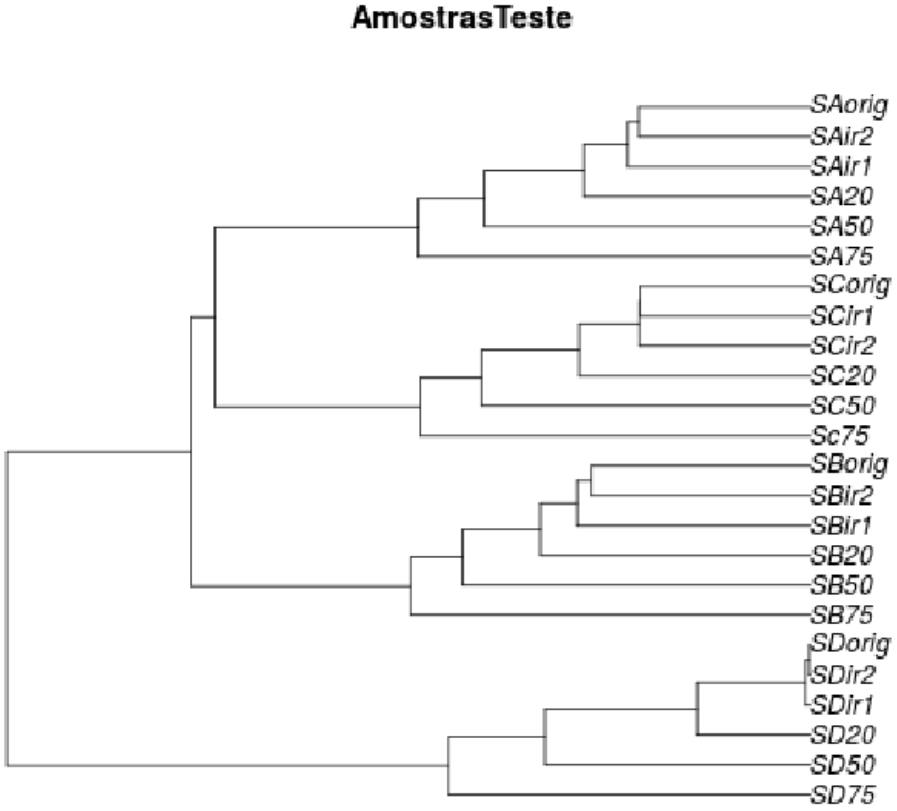

Figura 7: Cladograma gerado pela ferramenta desenvolvida, utilizando o conjunto de amostras descrito na Tabela 1 e o algoritmo de Boruvka.

Os genes selecionados para serem incorporados ao conjunto de amostras foram os que possuiam suas sequências completas e que estavam presentes em mais de duas espécies distintas. As sequências desses genes foram retiradas do *site* do Centro Nacional de Informações sobre Biotecnologia, do inglês *National Center for Biotechnology Information* (NCBI), que é uma seção da Biblioteca Nacional de Medicina dos Estados Unidos da América que hospeda dados da sequenciação de genomas, artigos de investigação biomédica e ferramentas da área da biotecnologia. Após definidas as amostras, a geração das árvores filogenéticas foi realizada entre as sequências de um mesmo gene presente em espécies distintas. No total foram colhidas sequências de 149 genes, listadas no Anexo A.

Cada um dos 149 genes foram testados com os três algoritmos implementados na ferramenta. O tempo total gasto pela ferramenta para criar as árvores filogenéticas para todos os genes, utilizando os três algoritmos foi de aproximadamente cinco horas e trinta e um minutos. A Tabela 2 mostra o tempo total gasto por cada algoritmo para executar o conjunto com 149 genes. Podemos verificar que os tempos foram praticamente iguais, com uma leve vantagem para o algoritmo de Boruvka.

	Boruvka	Kruskal	Prim
Tempo	1h 50m 7s	1h 50m 11s	1h 50m 50s

Tabela 2: Tempos de execução dos algoritmos de Boruvka, Kruskal e Prim para todos os genes.

Para exemplificar como foram os resultados gerados pela ferramenta, foi escolhido o gene `fam71f1` que está presente nas espécies *Bos Taurus*, *Canis Lupus*, *Homo Sapiens*, *Mus Musculus* e *Pan Troglodytes*. A árvore filogenética obtida pela ferramenta, para essas cinco sequências, está representada na Figura 8.

De acordo com o cladograma da Figura 8 observa-se que as espécies *Canis Lupus* e *Bos Taurus* assim como as espécies *Pan Troglodytes* e *Homo Sapiens*, são as que estão mais próximas evolutivamente. Podemos observar também que a espécie *Mus Musculus* é a mais distante de todas as outras.

A fim de verificar a qualidade das árvores filogenéticas geradas pela nossa ferramenta, utilizamos nesse trabalho uma medida estatística denominada correlação cofenética. A **correlação cofenética**

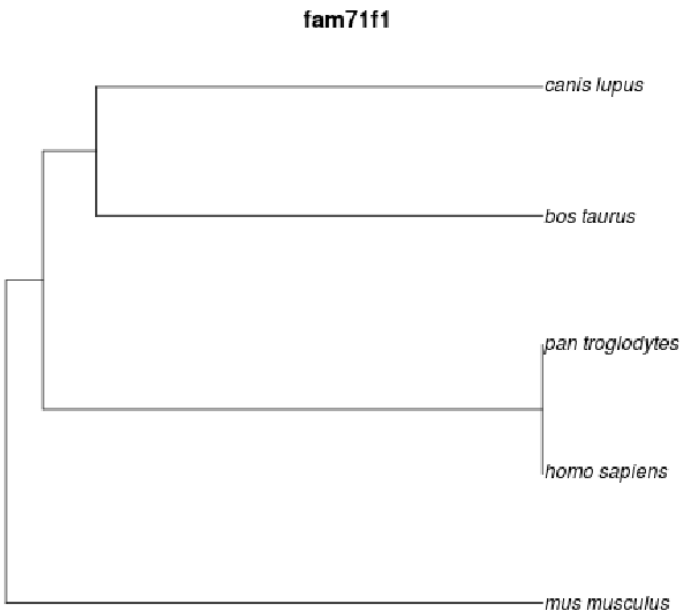

Figura 8: Árvore filogenética gerada pela ferramenta, criada com as sequências do gene `fam71f1`, utilizando o algoritmo de Boruvka.

mede o grau de ajuste entre a matriz de similaridade original e uma matriz resultante de algum método de agrupamento. No caso desse trabalho o método de agrupamento utilizado foram os algoritmos de obtenção de árvores geradoras de custo mínimo [7].

Foram utilizados dois métodos de correlação, a **correlação de Pearson** e a **correlação de Spearman**. A **correlação de Pearson** é um coeficiente que mede o grau de relação entre duas variáveis analisadas. Essa variável pode assumir valores entre -1 e 1. Quando o valor obtido é igual a 1, dizemos que houve uma correlação perfeita entre as duas variáveis, se o valor for igual a -1, significa que há uma correlação inversa, ou seja, se uma das variáveis aumenta a outra diminui. Finalmente, se o valor for igual a 0 significa que não há uma dependência linear entre as variáveis analisadas. A **correlação de Spearman** inicia seu cálculo substituindo os valores originais por seus índices relativos, considerando a ordem não decrescente de cada posição. Em seguida a correlação de Pearson é calculada sobre o conjunto de dados a partir desses índices ao invés dos valores dos atributos originais [19].

Mostramos na Tabela 3 os valores das correlações de Pearson e Spearman calculados a partir do cladograma da Figura 7.

	Pearson	Spearman
Boruvka	0,99256	0,9576899
Kruskal	0,99256	0,9576899
Prim	0,99256	0,9576899

Tabela 3: Correlações de Pearson e Spearman entre os cladogramas gerados pelos algoritmos para a criação de árvore geradora de custo mínimo e a similaridade entre as sequências artificiais.

Analisando a Tabela 3 foi possível verificar que o cladograma gerado está condizente com a similaridade calculada, confirmando o que foi analisado no cladograma da Figura 7, gerado a partir das sequências da Tabela 1.

Podemos avaliar o cladograma da Figura 8 verificando os valores das correlações apresentados na Tabela 4. Como analisado nos outros exemplos, os valores do cálculo das correlações de Pearson e Spearman estão próximos a 1, concluindo assim que o cladograma gerado está condizente com a similaridade entre as espécies.

Com relação às sequências dos 149 genes utilizados para testar a ferramenta, podemos ver a média das correlações de Pearson e Spearman na Tabela 5, e concluir que os cladogramas gerados estão condizentes com a similiaridade calculada entres as sequências já que os valores obtidos estão próximos de 1.

	Pearson	Spearman
Boruvka	0,99786	0,93744
Kruskal	0,99786	0,93744
Prim	0,99786	0,93744

Tabela 4: Correlações de Pearson e Spearman dos três algoritmos para a geração de árvore geradora de custo mínimo do gene fam71f1.

	Pearson	Spearman
Boruvka	0,9216054	0,895225
Kruskal	0,9216054	0,895225
Prim	0,9216054	0,895225

Tabela 5: Média das correlações de Pearson e Spearman dos três algoritmos para a criação de árvore geradora de custo mínimo para todos os genes.

6 Conclusão

O objetivo principal do nosso trabalho era criar uma ferramenta computacional para geração de árvores filogenéticas. Para alcançar tal objetivo, realizamos o alinhamento entre as sequências de interesse, duas a duas, e criamos um grafo completo onde os vértices correspondem às espécies analisadas e as arestas aos valores das similaridades entre essas espécies. Sobre esse grafo completo era aplicado um algoritmo para a criação de uma árvore geradora de custo mínimo, a qual dá origem ao cladograma que representa a árvore filogenética.

Na Seção 5 foram apresentados os resultados obtidos com os testes executados sobre nossa ferramenta, utilizando os três algoritmos para criação de árvore geradora de custo mínimo implementados. Os resultados obtidos pela ferramenta foram sempre os mesmos, independente do algoritmo utilizado. A única diferença notada entre os três algoritmos foi no tempo de execução, onde o algoritmo de Boruvka se mostrou ligeiramente mais rápido que os demais. Com relação à qualidade das árvores geradas, utilizamos duas medidas estatísticas para a avalição dos resultados. Através dessas medidas podemos notar que a ferramenta obteve bons resultados para a grande parte dos genes avaliados.

Durante nosso estudo, pudemos perceber a possibilidade de melhorar o desempenho da ferramenta utilizando algumas estruturas de dados mais avançadas. Por exemplo, poderiam ser utillizadas estruturas como heap de Fibonacci e conjunto disjuntos para melhorar o tempo de execução das ferramentas ou então utilizar uma técnica de alinhamento múltiplo de sequências, ao invés do alinhamento entre duas sequências, o que poderia nos dar mais informações sobre a similaridade entre uma sequência específica e todas as outras analisadas.

Referências

[1] ALBERTS, Bruce; JOHNSON, Alexander; LEWIS, Julian; RAFF, Martin; ROBERTS, Keith; WALKER, Peter. Biologia Molecular da Celula. Tradução Anne D. Villela, Ardala Elisa Breda Andrade, Carlos Alexandre Sanches Ferreira,Carlos Termignoni, Cláudia Paiva Nunes, Cristopher Zandoná Schneider, Denise C. Machado, Diógenes Santiago Santos, Gaby Renard, Giancarlo Pasquali, Heique Marlis Bogdawa, Jacqueline Moraes Cardone, José Arthur B. Chies, José Eduardo Nunes Saconni, Rosane Machado Scheibe, Rui Fernando Felix Lopes, Sandra Estrazulas Farias. 5ª ed. Porto Alegre, Artmed (2010).

[2] AMORIM, Dalton de Souza. Fundamentos de Sistemática Filogenetica (1º edição), Editora Holos, 2002.

[3] AURICCHIO, Paulo; NERA, Paulo N. História evolutiva de primates: Análise filogenética de Callicebus Thomas, 1993 (Primates - Pitheciidae - Callicebinae).Instituto de Biociências da Universidade de São Paulo. 2005.

[4] BONDY, John A.; MURTY, U. S. R. Graph theory with applications. North-holland. University of Waterloo. 1982.

[5] BRITO, Rogério Theodoro. Alinhamento de Sequências Biológicas. USP, São Paulo, 2003.

[6] BROOKSHEAR, J. Glenn. Ciência da Computação, Uma visão abrangente. 7ª Edição, Editora Bookman, São Paulo, 2003.

[7] BUSSAB, W. DE O.; MIAZAKI, E.S.; ANDRADE, D. F. Introdução à análise de agrupamentos. São Paulo: Associação Brasileira de Estatística, 1990. 105p.

[8] CARDOSO, Domingos Moreira. Teoria dos Grafos e Aplicações. Departamento de Matemática da Universidade de Aveiro. Mestrado em Matemática, 2005.

[9] CHING, Francis D. K.; ECKLER, James F. Bioquímica. 4ª Edição, Editora Artmed, 2013;

[10] CORMEN, Thomas H.;LEISESON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. Algoritmos Teoria e Pratrica. Tradução Arlete Simille Marques. Rio de Janeiro: Elsevier, 2012.

[11] COX, Michael M.; DOUDNA, Jennifer A.; O'DONNELL, Michael. Biologia Molecular: Princípios e Técnicas. Editora Artmed, 2012.

[12] DARWIN, C."On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life". John Murray. Londres. 1859.

[13] DELSUC, F.; BRINKMANN, H.; PHILIPPE, H. "Phylogenomics and the Reconstruction of the Tree of Life". Nature Reviews|Genetics. Nature Publishing Group. vol 6, pp. 361-375, 2005.

[14] FELSENSTEIN, J. Inferring phylogenies. Sunderlan, Massachussts: Sinauer, 2004.

[15] FEOFILOFF, P.; KOHAYAKAWA, Y.; WAKABAYASHI, Y. Uma Introdução Sucinta à Teoria dos Grafos, Segunda Bienal da Sociedade Brasileira de Matemática, Salvador, 2011.

[16] GOLDBARG, Marco Cesar; LUNA, Henrique P. L. Otimização Combinatória e Programação Linear: Modelos e Algoritmos. Elsevier, 2ª ed., Rio de Janeiro, 2005.

[17] GUSFIELD, Dan. Algorithms on strings, trees and sequences: computer science and computational biology. Cambridge University Press. 2009.

[18] LINDER, C.R.; WARNOW, T. "An Overview of Phylogeny Reconstruction". "Handbook of Computational Molecular Biology". Ed. Avaru, S. Chapman & Hall/CRC. Cap. 19,pp. 1-34, 2006.

[19] METZ, Jean. "Interpretação de clusters gerados por algoritmos de *clustering* hierárquico". ICMC-USP, São Carlos, 2006.

[20] MEYER, Diogo; EL-HANI, Charbel N. "Evolução: O Sentido da Biologia". Editora Unesp, Série Evolução. Cap. 01, pag 15, 2006

[21] NEEDLEMAN, S. B., and WUNSCH, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology 48 (1970), 443-453.

[22] NOBRE, Ricardo Holanda. Paralelismo como solução para redução de complexidade de problemas combinatoriais. Departamento de Estatística e Computação, Universidade Estadual do Ceará (2011).

[23] About R. disponível em <http://www.r-project.org>, acessado em 03/11/2014.

[24] ROBERTIS, E. M. F. D., and HIB, J. Bases da Biologia Celular e Molecular,4 ed. Guanabara Koogan, 2006.

[25] SADAVA, David; HELLER, Craig; ORIANS, Gordon H.; PURVES Willian K.; HILLIS, David M. Vida: A Ciência da Biologia - Volume 1: Célula e Hereditariedade. 8ª Edição, Editora Artmed, 2014;

[26] SETUBAL, João; MEIDANIS, João. Introduction to Computational Molecular Biology. University of Campinas, Brazil. PWS Publishing Company (1997).

[27] The ENCODE Project Consortium. The ENCODE (encyclopedia of DNA elements) project. Science 306, 5696 (2004), 636?640.

Anexo A Amostras utilizadas no teste

São listados na Tabela A.1 abaixo, os 149 genes utilizados no teste da ferramenta desenvolvida nesse trabalho. Para cada gene, está descrito a quantidade de espécies que o possuem e quais são essas espécies.

Genes	**Qtd**	**Espécies**
a1bg	6	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes* e *Rattus norvegicus*
alg10b	4	*Homo sapiens*, *Macaca mulatta*, *Mus musculus* e *Pan troglodytes*
ankrd10	5	*Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes* e *Xenopus tropicalis*
arhgdig	8	*Bos taurus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus* e *Xenopus tropicalis*
ascc2	7	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Danio rerio* e *Xenopus tropicalis*
ascl2	7	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes* e *Rattus norvegicus*
bad	6	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes* e *Rattus norvegicus*
bet1	12	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Drosophila melanogaster*, *Gallus gallus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Saccharomyces cerevisiae* e *Xenopus tropicalis*
bgn	7	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Homo sapiens*, *Mus musculus*, *Rattus norvegicus* e *Xenopus tropicalis*
c21orf59	4	*Danio rerio*, *Gallus gallus*, *Homo sapiens* e *Xenopus tropicalis*
cav1	9	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Xenopus tropicalis*
ccdc157	9	*Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Xenopus tropicalis*
ccdc88b	6	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Homo sapiens*, *Mus musculus* e *Pan troglodytes*
ccl5	7	*Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes* e *Rattus norvegicus*
cdx2	6	*Canis lupus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Rattus norvegicus* e *Xenopus tropicalis*
chmp2a	8	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus* e *Xenopus tropicalis*
cldn12	9	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes* e *Xenopus tropicalis*
csf2	7	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes* e *Rattus norvegicus*
ctgf	8	*Bos taurus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus* e *Xenopus tropicalis*
ctsd	8	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus* e *Rattus norvegicus*
ddx18	9	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Rattus norvegicus* e *Xenopus tropicalis*
ddx43	7	*Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus* e *Xenopus tropicalis*
decr2	5	*Danio rerio*, *Homo sapiens*, *Mus musculus*, *Rattus norvegicus* e *Xenopus tropicalis*
dnajc4	8	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes* e *Rattus norvegicus*
continua na próxima página		

Gene	**Qtd**	**Espécies**
dppa5	4	*Canis lupus*, *Homo sapiens*,*Pan troglodytes* e *Rattus norvegicus*
drg1	9	*Arabidopsis thaliana*, *Bos taurus*, *Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes* e *Rattus norvegicus*
dusp18	9	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus* e *Xenopus tropicalis*
eef1a1	8	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes* e *Rattus norvegicus*
esrra	7	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes* e *Rattus norvegicus*
fam71f1	5	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Mus musculus* e *Pan troglodytes*
fkbp2	8	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes* e *Xenopus tropicalis*
frs3	9	*Arabidopsis thaliana*, *Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus* e *Xenopus tropicalis*
gabrq	6	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus* e *Rattus norvegicus*
gal3st1	6	*Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, e *Xenopus tropicalis*
gas2l2	7	*Bos taurus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus* e *Xenopus tropicalis*
gdf9	8	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes* e *Rattus norvegicus*
gng11	8	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes* e *Rattus norvegicus*
gngt1	6	*Danio rerio*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus* e *Xenopus tropicalis*
gpr137	7	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes* e *Rattus norvegicus*
gsx1	8	*Canis lupus*, *Danio rerio*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus* e *Xenopus tropicalis*
hba1	4	*Bos taurus*, *Homo sapiens*, *Rattus norvegicus* e *Xenopus tropicalis*
hba2	4	*Homo sapiens*, *Macaca mulatta*, *Rattus norvegicus* e *Xenopus tropicalis*
hbb	5	*Bos taurus*, *Homo sapiens*, *Macaca mulatta*, *Pan troglodytes* e *Rattus norvegicus*
hbe1	7	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Macaca mulatta*, *Rattus norvegicus* e *Xenopus tropicalis*
hbg1	3	*Gallus gallus*, *Homo sapiens* e *Rattus norvegicus*
hbg2	4	*Gallus gallus*, *Homo sapiens*, *Macaca mulatta* e *Xenopus tropicalis*
hbq1	3	*Bos taurus*, *Homo sapiens* e *Macaca mulatta*
hbz	5	*Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Rattus norvegicus* e *Xenopus tropicalis*
hoxa2	9	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus* e *Xenopus tropicalis*
hoxa4	7	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes* e *Xenopus tropicalis*
hoxa5	8	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus* e *Xenopus tropicalis*
hoxa6	9	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus* e *Xenopus tropicalis*
hoxa7	7	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes* e *Xenopus tropicalis*
continua na próxima página		

Gene	**Qtd**	**Espécies**
il13	6	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes* e *Rattus norvegicus*
il3	5	*Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes* e *Rattus norvegicus*
il5	7	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes* e *Rattus norvegicus*
inpp5j	7	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes* e *Xenopus tropicalis*
ins	6	*Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Pan troglodytes* e *Xenopus tropicalis*
insig2	7	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes* e *Rattus norvegicus*
irf1	9	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus* e *Xenopus tropicalis*
itfg3	5	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens* e *Mus musculus*
kcnk4	6	*Canis lupus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus* e *Xenopus tropicalis*
leap2	6	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*
lep	7	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*
lif	5	*Bos taurus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Rattus norvegicus*
lrrc4	8	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Xenopus tropicalis*
lyzl6	5	*Bos taurus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*
mdfi	8	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Xenopus tropicalis*
mettl2b	4	*Bos taurus*, *Homo sapiens*, *Pan troglodytes*, *Rattus norvegicus*
mier3	8	*Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Xenopus tropicalis*
mmp26	4	*Canis lupus*, *Macaca mulatta*, *Homo sapiens*, *Pan troglodytes*
mrpl23	12	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Drosophila melanogaster*, *Gallus gallus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Rattus norvegicus*, *Saccharomyces cerevisiae*, *Schizosaccharomyces pombe*, *Xenopus tropicalis*
mzf1	4	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Mus musculus*
nipal1	7	*Bos taurus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Xenopus tropicalis*
nme4	7	*Canis lupus*, *Danio rerio*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Xenopus tropicalis*
nr2e1	7	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*
nsdhl	6	*Bos taurus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Xenopus tropicalis*
ooep	6	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*
or51g2	3	*Canis lupus*, *Homo sapiens*, *Macaca mulatta*
or51m1	5	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Macaca mulatta*
or51q1	3	*Bos taurus*, *Canis lupus*, *Homo sapiens*
or51s1	3	*Bos taurus*, *Homo sapiens*, *Macaca mulatta*
continua na próxima página		

Gene	**Qtd**	**Espécies**
or51t1	3	*Bos taurus*, *Homo sapiens*, *Rattus norvegicus*
or52h1	3	*Bos taurus*, *Pan troglodytes*, *Homo sapiens*
or52j3	3	*Bos taurus*, *Canis lupus*, *Homo sapiens*
osm	6	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*
ostm1	7	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Xenopus tropicalis*
pdia2	6	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Mus musculus*, *Rattus norvegicus*, *Xenopus tropicalis*
pdk4	8	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Xenopus tropicalis*
pdx1	10	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Gallus gallus*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Saccharomyces cerevisiae*, *Xenopus tropicalis*
pes1	6	*Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Rattus norvegicus*
pex12	11	*Arabidopsis thaliana*, *Bos taurus*, *Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Saccharomyces cerevisiae*
pla2g3	6	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*
pnma3	5	*Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*
pnma5	4	*Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Rattus norvegicus*
polr3k	7	*Bos taurus*, *Danio rerio*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Xenopus tropicalis*
pon1	8	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*
pon3	8	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*
ppp1r14b	7	*Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Xenopus tropicalis*
ppp1r3a	7	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*
prickle4	4	*Canis lupus*, *Homo sapiens*, *Mus musculus*, *Rattus norvegicus*
rasl10b	6	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Rattus norvegicus*
rbm28	9	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Xenopus tropicalis*
rps5	11	*Arabidopsis thaliana*, *Bos taurus*, *Canis lupus*, *Danio rerio*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Saccharomyces cerevisiae*, *Xenopus tropicalis*
samd9	4	*Bos taurus*, *Homo sapiens*, *Macaca mulatta*, *Pan troglodytes*
samd9l	5	*Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Xenopus tropicalis*
sec14l3	6	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*
sec14l4	5	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Mus musculus*, *Rattus norvegicus*
selm	8	*Canis lupus*, *Danio rerio*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Xenopus tropicalis*

continua na próxima página

Gene	**Qtd**	**Espécies**
shroom1	6	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*
skap2	6	*Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*
slc22a11	3	*Canis lupus*, *Homo sapiens*, *Macaca mulatta*
slc22a5	5	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Mus musculus*, *Rattus norvegicus*
slc27a5	3	*Bos taurus*, *Homo sapiens*, *Mus musculus*
slc35e4	8	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Rattus norvegicus*, *Xenopus tropicalis*
slc4a3	7	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*
slfn13	4	*Homo sapiens*, *Macaca mulatta*, *Pan troglodytes*, *Rattus norvegicus*
slfn14	5	*Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*
snrnp25	9	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Xenopus tropicalis*
snx19	7	*Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*
spp2	7	*Arabidopsis thaliana*, *Bos taurus*, *Canis lupus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Saccharomyces cerevisiae*
steap1	9	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Xenopus tropicalis*
stip1	7	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Homo sapiens*, *Mus musculus*, *Rattus norvegicus*, *Xenopus tropicalis*
syt8	6	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Rattus norvegicus*
taf15	4	*Arabidopsis thaliana*, *Danio rerio*, *Homo sapiens*, *Mus musculus*
tbc1d10a	8	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Rattus norvegicus*, *Xenopus tropicalis*
tcn2	7	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Rattus norvegicus*
tfpi2	10	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Xenopus tropicalis*
tnni2	7	*Bos taurus*, *Canis lupus*, *Gallus gallus*, *Homo sapiens*, *Mus musculus*, *Rattus norvegicus*, *Xenopus tropicalis*
tomm6	7	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*
trex2	5	*Canis lupus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*
trim28	3	*Homo sapiens*, *Mus musculus*, *Rattus norvegicus*
ube2m	7	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*, *Xenopus tropicalis*
ubqln3	7	*Bos taurus*, *Canis lupus*, *Homo sapiens Macaca mulatta*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*
ubqlnl	6	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*, *Rattus norvegicus*
vegfb	4	*Bos taurus*, *Canis lupus*, *Mus musculus*, *Rattus norvegicus*
zbtb45	5	*Canis lupus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Rattus norvegicus*
zfp92	4	*Canis lupus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*
continua na próxima página		

Gene	**Qtd**	**Espécies**
zmat5	7	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Macaca mulatta*, *Mus musculus*, *Xenopus tropicalis*
znf132	5	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Macaca mulatta*, *Pan troglodytes*
znf135	3	*Bos taurus*, *Homo sapiens*, *Pan troglodytes*
znf324	3	*Bos taurus*, *Canis lupus*, *Homo sapiens*
znf418	3	*Bos taurus*, *Canis lupus*, *Homo sapiens*
znf446	3	*Bos taurus*, *Canis lupus*, *Homo sapiens*
znf584	5	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Macaca mulatta*, *Pan troglodytes*
znf622	8	*Bos taurus*, *Canis lupus*, *Danio rerio*, *Gallus gallus*, *Homo sapiens*, *Macaca mulatta*, *Pan troglodytes*, *Xenopus tropicalis*
znf8	3	*Canis lupus*, *Homo sapiens*, *Macaca mulatta*
znf814	3	*Canis lupus*, *Homo sapiens*, *Pan troglodytes*
zscan22	4	*Canis lupus*, *Homo sapiens*, *Mus musculus*, *Pan troglodytes*
zscan4	4	*Bos taurus*, *Canis lupus*, *Homo sapiens*, *Macaca mulatta*

Tabela A.1: Listagem dos genes e das espécies que possuem esse gene, utilizados no teste da ferramenta desenvolvida. Na coluna Gene estão os nomes dos genes estudados; na coluna Qtd, a quantidade de sequências coletadas para o referido gene e, por fim, na coluna Espécies estão descritas as espécies das quais foram coletadas as sequências.

Anexo B Correlações

Na Tabela B.1 são listadas as correlações de Pearson e Spearman calculadas para cada gene utilizado como entrada para a ferramenta desenvolvida. Vale lembrar que, para cada gene, os valores da correlação de Pearson são iguais independente do algoritimo para geração da árvore que foi utilizado. A mesma coisa ocorre com relação à correlação de Spearman.

A última linha da Tabela B.1 possui a média das correlações considerando todos os genes.

Gene	**Boruvka**		**Kruskal**		**Prim**	
	Pearson	**Spearman**	**Pearson**	**Spearman**	**Pearson**	**Spearman**
a1bg	0.94144	0.94868	0.94144	0.94868	0.94144	0.94868
alg10b	0.87238	0.92582	0.87238	0.92582	0.87238	0.92582
ankrd10	0.91860	0.95346	0.91860	0.95346	0.91860	0.95346
arhgdig	0.97261	0.92963	0.97261	0.92963	0.97261	0.92963
ascc2	0.84174	0.85333	0.84174	0.85333	0.84174	0.85333
ascl2	0.99959	0.93319	0.99959	0.93319	0.99959	0.93319
bad	0.96055	0.84175	0.96055	0.84175	0.96055	0.84175
bet1	0.94830	0.95553	0.94830	0.95553	0.94830	0.95553
bgn	0.90580	0.93801	0.90580	0.93801	0.90580	0.93801
c21orf59	0.96626	0.92582	0.96626	0.92582	0.96626	0.92582
cav1	0.91799	0.96068	0.91799	0.96068	0.91799	0.96068
ccdc157	0.83911	0.90322	0.83911	0.90322	0.83911	0.90322
ccdc88b	0.99806	0.96362	0.99806	0.96362	0.99806	0.96362
ccl5	0.92531	0.95145	0.92531	0.95145	0.92531	0.95145
cdx2	0.98610	0.96362	0.98610	0.96362	0.98610	0.96362
chmp2a	0.88428	0.90498	0.88428	0.90498	0.88428	0.90498
cldn12	0.76941	0.80082	0.76941	0.80082	0.76941	0.80082
csf2	0.96248	0.90058	0.96248	0.90058	0.96248	0.90058
ctgf	0.98736	0.96515	0.98736	0.96515	0.98736	0.96515
ctsd	0.93282	0.70331	0.93282	0.70331	0.93282	0.70331
ddx18	0.88848	0.95383	0.88848	0.95383	0.88848	0.95383
ddx43	0.83772	0.89072	0.83772	0.89072	0.83772	0.89072
decr2	0.68415	0.66487	0.68415	0.66487	0.68415	0.66487
dnajc4	0.99872	0.95057	0.99872	0.95057	0.99872	0.95057
dppa5	0.93107	0.84515	0.93107	0.84515	0.93107	0.84515
drg1	0.67009	0.83927	0.67009	0.83927	0.67009	0.83927
dusp18	0.88966	0.92742	0.88966	0.92742	0.88966	0.92742
eef1a1	0.99786	0.86043	0.99786	0.86043	0.99786	0.86043
esrra	0.98032	0.95720	0.98032	0.95720	0.98032	0.95720
fam71f1	0.99786	0.93744	0.99786	0.93744	0.99786	0.93744
fkbp2	0.93573	0.97968	0.93573	0.97968	0.93573	0.97968
frs3	0.87905	0.93274	0.87905	0.93274	0.87905	0.93274
gabrq	0.99446	0.88950	0.99446	0.88950	0.99446	0.88950
gal3st1	0.92599	0.91107	0.92599	0.91107	0.92599	0.91107
gas2l2	0.99383	0.86483	0.99383	0.86483	0.99383	0.86483
gdf9	0.74930	0.88699	0.74930	0.88699	0.74930	0.88699
gng11	0.98626	0.96717	0.98626	0.96717	0.98626	0.96717
gngt1	0.94400	0.96825	0.94400	0.96825	0.94400	0.96825
gpr137	0.99422	0.97036	0.99422	0.97036	0.99422	0.97036
gsx1	0.89182	0.91482	0.89182	0.91482	0.89182	0.91482
hba1	0.96312	0.92582	0.96312	0.92582	0.96312	0.92582
hba2	0.99992	0.92582	0.99992	0.92582	0.99992	0.92582
hbb	0.96461	0.90897	0.96461	0.90897	0.96461	0.90897

continua na próxima página

Gene	Boruvka		Kruskal		Prim	
	Pearson	**Spearman**	**Pearson**	**Spearman**	**Pearson**	**Spearman**
hbe1	0.62694	0.51110	0.62694	0.51110	0.62694	0.51110
hbg1	0.95789	0.86603	0.95789	0.86603	0.95789	0.86603
hbg2	0.99789	0.92582	0.99789	0.92582	0.99789	0.92582
hbq1	0.99065	0.86603	0.99065	0.86603	0.99065	0.86603
hbz	0.99963	0.95346	0.99963	0.95346	0.99963	0.95346
hoxa2	0.98112	0.92584	0.98112	0.92584	0.98112	0.92584
hoxa4	0.99545	0.97701	0.99545	0.97701	0.99545	0.97701
hoxa5	0.91476	0.94876	0.91476	0.94876	0.91476	0.94876
hoxa6	0.93991	0.93397	0.93991	0.93397	0.93991	0.93397
hoxa7	0.92935	0.93246	0.92935	0.93246	0.92935	0.93246
il13	0.63710	0.78387	0.63710	0.78387	0.63710	0.78387
il3	0.97928	0.88420	0.97928	0.88420	0.97928	0.88420
il5	0.98379	0.90038	0.98379	0.90038	0.98379	0.90038
inpp5j	0.92891	0.96935	0.92891	0.96935	0.92891	0.96935
ins	0.77540	0.83997	0.77540	0.83997	0.77540	0.83997
insig2	0.83669	0.95528	0.83669	0.95528	0.83669	0.95528
irf1	0.94461	0.92635	0.94461	0.92635	0.94461	0.92635
itfg3	0.94193	0.86364	0.94193	0.86364	0.94193	0.86364
kcnk4	0.99665	0.94868	0.99665	0.94868	0.99665	0.94868
leap2	0.99338	0.80595	0.99338	0.80595	0.99338	0.80595
lep	0.97742	0.81398	0.97742	0.81398	0.97742	0.81398
lif	0.69881	0.62711	0.69880	0.62711	0.69880	0.62711
lrrc4	0.99993	0.97577	0.99993	0.97577	0.99993	0.97577
lyzl6	0.99149	0.93744	0.99149	0.93744	0.99149	0.93744
mdfi	0.98027	0.91612	0.98027	0.91612	0.98027	0.91612
mettl2b	0.98758	0.84515	0.98758	0.84515	0.98758	0.84515
mier3	0.85609	0.91766	0.85609	0.91766	0.85609	0.91766
mmp26	0.99980	0.92582	0.99979	0.92582	0.99979	0.92582
mrpl23	0.93889	0.94736	0.93889	0.94736	0.93889	0.94736
mzf1	0.90316	0.92582	0.90316	0.92582	0.90316	0.92582
nipal1	0.93334	0.91343	0.93334	0.91343	0.93334	0.91343
nme4	0.90284	0.96970	0.90284	0.96970	0.90284	0.96970
nr2e1	0.95138	0.97701	0.95138	0.97701	0.95138	0.97701
nsdhl	0.83708	0.88439	0.83708	0.88439	0.83708	0.88439
ooep	0.91619	0.75247	0.91619	0.75247	0.91619	0.75247
or51g2	0.66765	0.86603	0.66765	0.86603	0.66765	0.86603
or51m1	0.99428	0.93744	0.99428	0.93744	0.99428	0.93744
or51q1	0.99632	0.86603	0.99632	0.86603	0.99632	0.86603
or51s1	0.99676	0.86603	0.99676	0.86603	0.99676	0.86603
or51t1	0.99988	0.86603	0.99988	0.86603	0.99988	0.86603
or52h1	0.65604	0.86603	0.65604	0.86603	0.65604	0.86603
or52j3	0.99999	0.86603	0.99999	0.86603	0.99999	0.86603
osm	0.97002	0.95534	0.97002	0.95534	0.97002	0.95534
ostm1	0.80220	0.85694	0.80220	0.85694	0.80220	0.85694
pdia2	0.99754	0.90370	0.99754	0.90370	0.99754	0.90370
pdk4	0.99089	0.87733	0.99089	0.87733	0.99089	0.87733
pdx1	0.98814	0.95952	0.98814	0.95952	0.98814	0.95952
pes1	0.82007	0.88023	0.82007	0.88023	0.82007	0.88023
pex12	0.79140	0.84120	0.79140	0.84120	0.79140	0.84120
pla2g3	0.99876	0.96825	0.99876	0.96825	0.99876	0.96825
pnma3	0.83331	0.78184	0.83331	0.78184	0.83331	0.78184

continua na próxima página

Gene	Boruvka		Kruskal		Prim	
	Pearson	**Spearman**	**Pearson**	**Spearman**	**Pearson**	**Spearman**
pnma5	0.90219	0.92582	0.90219	0.92582	0.90219	0.92582
polr3k	0.81362	0.85515	0.81362	0.85515	0.81362	0.85515
pon1	0.97430	0.95426	0.97429	0.95426	0.97429	0.95426
pon3	0.96660	0.71328	0.96659	0.71328	0.96659	0.71328
ppp1r14b	0.93739	0.97535	0.93739	0.97535	0.93739	0.97535
ppp1r3a	0.98806	0.91420	0.98806	0.91420	0.98806	0.91420
prickle4	0.98264	0.84515	0.98264	0.84515	0.98264	0.84515
rasl10b	0.97843	0.90736	0.97843	0.90736	0.97843	0.90736
rbm28	0.95517	0.97781	0.95517	0.97781	0.95517	0.97781
rps5	0.68942	0.82913	0.68942	0.82913	0.68942	0.82913
samd9	0.99999	0.92582	0.99999	0.92582	0.99999	0.92582
samd9l	0.85944	0.88420	0.85944	0.88420	0.85944	0.88420
sec14l3	0.75286	0.84139	0.75286	0.84139	0.75286	0.84139
sec14l4	0.78349	0.93744	0.78349	0.93744	0.78349	0.93744
selm	0.91726	0.95448	0.91726	0.95448	0.91726	0.95448
shroom1	0.94410	0.92798	0.94410	0.92798	0.94410	0.92798
slc22a11	0.99989	0.86603	0.99989	0.86603	0.99989	0.86603
slc22a5	0.98715	0.93744	0.98715	0.93744	0.98715	0.93744
slc27a5	0.94955	0.86603	0.94955	0.86603	0.94955	0.86603
slc35e4	0.85143	0.88885	0.85143	0.88885	0.85143	0.88885
slc4a3	0.99066	0.94738	0.99066	0.94738	0.99066	0.94738
slfn13	0.64621	0.61721	0.64621	0.61721	0.64621	0.61721
slfn14	0.97054	0.83400	0.97054	0.83400	0.97054	0.83400
snrnp25	0.99574	0.84790	0.99574	0.84790	0.99574	0.84790
snx19	0.99574	0.94791	0.99574	0.94791	0.99574	0.94791
spp2	0.98919	0.92454	0.98919	0.92454	0.98919	0.92454
steap1	0.94948	0.71305	0.94948	0.71305	0.94948	0.71305
stip1	0.87193	0.90022	0.87193	0.90022	0.87193	0.90022
syt8	0.93387	0.87833	0.93387	0.87833	0.93387	0.87833
taf15	0.83807	0.92582	0.83807	0.92582	0.83807	0.92582
tbc1d10a	0.93421	0.93485	0.93420	0.93485	0.93420	0.93485
tcn2	0.95914	0.97535	0.95914	0.97535	0.95914	0.97535
tfpi2	0.97976	0.97004	0.97976	0.97004	0.97976	0.97004
tnni2	0.92134	0.95890	0.92134	0.95890	0.92134	0.95890
tomm6	0.97743	0.93037	0.97743	0.93037	0.97743	0.93037
trex2	0.94905	0.95346	0.94905	0.95346	0.94905	0.95346
trim28	0.98508	0.86603	0.98508	0.86603	0.98508	0.86603
ube2m	0.97118	0.97000	0.97118	0.97000	0.97118	0.97000
ubqln3	0.89639	0.89903	0.89639	0.89903	0.89639	0.89903
ubqlnl	0.99973	0.94868	0.99973	0.94868	0.99973	0.94868
vegfb	0.98041	0.84515	0.98041	0.84515	0.98041	0.84515
zbtb45	0.92287	0.88420	0.92287	0.88420	0.92287	0.88420
zfp92	0.93951	0.92582	0.93951	0.92582	0.93951	0.92582
zmat5	0.78618	0.85360	0.78618	0.85360	0.78618	0.85360
znf132	0.97096	0.95346	0.97096	0.95346	0.97096	0.95346
znf135	0.99993	0.86603	0.99993	0.86603	0.99993	0.86603
znf324	0.93337	0.86603	0.93337	0.86603	0.93337	0.86603
znf418	0.97090	0.86603	0.97090	0.86603	0.97090	0.86603
znf446	0.99721	0.86603	0.99721	0.86603	0.99721	0.86603
znf584	0.94986	0.92168	0.94986	0.92168	0.94986	0.92168
znf622	0.91300	0.94704	0.91299	0.94704	0.91299	0.94704

continua na próxima página

Gene	Boruvka		Kruskal		Prim	
	Pearson	**Spearman**	**Pearson**	**Spearman**	**Pearson**	**Spearman**
znf8	0.94369	0.86603	0.94369	0.86603	0.94369	0.86603
znf814	0.96484	0.86603	0.96484	0.86603	0.96484	0.86603
zscan22	0.99900	0.84515	0.99900	0.84515	0.99900	0.84515
zscan4	0.70248	0.84515	0.70248	0.84515	0.70248	0.84515
Médias	**0.92161**	**0.89523**	**0.92161**	**0.89523**	**0.92161**	**0.89523**

Tabela B.1: Listagem dos valores das correlações de Pearson e Spearman, calculadas a partir da execução da nossa ferramenta utilizando os algoritmos de Boruvka, Kruskal e Prim, para todos os genes do conjunto de testes. Na coluna Genes são listados os nomes dos genes utilizados. Nas seis próximas colunas são mostradas, alternadamente, as correlações de Pearson e Spearman para as execuções da ferramenta feitas utilizando os algoritmos de Boruvka, Kruskal e Prim, respectivamente.

Anexo C Tempos de Execução

São listados na Tabela C.1 abaixo, os tempos que a ferramenta desenvolvida levou para gerar a árvore filogenética para cada espécie do conjunto de testes em relação ao algoritmo de árvore geradora de custo mínimo utilizado.

Podemos observar que o tempo gasto pela ferramenta, independente do algoritmo utilizado para gerar a árvore filogenética é semelhante para os três algoritmos.

Gene	**Boruvka**	**Kruskal**	**Prim**
a1bg	0m07.066s	0m06.993s	0m07.040s
alg10b	0m04.781s	0m04.767s	0m04.751s
ankrd10	1m54.243s	1m56.607s	1m57.583s
arhgdig	0m28.368s	0m28.418s	0m28.587s
ascc2	5m05.565s	5m08.081s	5m04.115s
ascl2	0m04.049s	0m04.080s	0m04.057s
bad	0m24.661s	0m24.422s	0m24.966s
bet1	1m09.837s	1m13.257s	1m10.456s
bgn	1m38.111s	1m35.652s	1m38.264s
c21orf59	0m15.032s	0m14.974s	0m15.476s
cav1	5m48.049s	5m51.259s	5m50.625s
ccdc157	1m30.734s	1m33.334s	1m28.988s
ccdc88b	1m35.202s	1m35.822s	1m27.990s
ccl5	0m08.133s	0m07.951s	0m08.100s
cdx2	0m08.353s	0m08.321s	0m08.239s
chmp2a	0m03.306s	0m03.404s	0m03.360s
cldn12	0m35.317s	0m35.417s	0m35.399s
csf2	0m01.819s	0m01.831s	0m01.821s
ctgf	0m03.255s	0m03.364s	0m03.373s
ctsd	0m37.020s	0m36.952s	0m37.174s
ddx18	1m38.633s	1m39.577s	1m41.871s
ddx43	2m55.665s	2m40.340s	2m38.937s
decr2	0m14.757s	0m14.592s	0m15.053s
dnajc4	0m10.590s	0m10.549s	0m10.125s
dppa5	0m00.604s	0m00.655s	0m00.611s
drg1	2m06.046s	2m07.965s	2m07.428s
dusp18	0m26.956s	0m27.291s	0m28.174s
eef1a1	0m16.130s	0m15.860s	0m15.660s
esrra	0m34.512s	0m33.259s	0m33.767s
fam71f1	0m54.556s	0m56.435s	0m55.945s
fkbp2	0m04.446s	0m04.384s	0m04.401s
frs3	1m06.797s	1m09.222s	1m07.155s
gabrq	1m29.110s	1m31.180s	1m29.774s
gal3st1	0m44.228s	0m44.450s	0m44.304s
gas2l2	0m21.927s	0m21.383s	0m21.559s
gdf9	0m09.928s	0m10.165s	0m09.933s
gng11	0m06.458s	0m06.466s	0m06.381s
gngt1	0m07.130s	0m07.351s	0m07.392s
gpr137	0m08.328s	0m08.279s	0m08.193s
gsx1	0m01.249s	0m01.289s	0m01.246s
hba1	0m00.495s	0m00.508s	0m00.500s
hba2	0m00.494s	0m00.525s	0m00.506s
hbb	0m00.683s	0m00.693s	0m00.692s
hbe1	0m00.924s	0m00.950s	0m00.942s
continua na próxima página			

Gene	Boruvka	Kruskal	Prim
hbg1	0m00.500s	0m00.522s	0m00.503s
hbg2	0m00.564s	0m00.575s	0m00.574s
hbq1	0m00.508s	0m00.509s	0m00.483s
hbz	0m01.139s	0m01.164s	0m01.151s
hoxa2	0m02.549s	0m02.585s	0m02.585s
hoxa4	0m01.428s	0m01.457s	0m01.424s
hoxa5	0m03.057s	0m03.069s	0m03.064s
hoxa6	0m17.107s	0m16.612s	0m16.560s
hoxa7	0m03.765s	0m03.996s	0m03.740s
il13	0m01.861s	0m01.894s	0m01.891s
il3	0m00.973s	0m01.007s	0m00.969s
il5	0m04.149s	0m04.215s	0m04.158s
inpp5j	0m35.744s	0m36.807s	0m36.500s
ins	0m01.587s	1m47.003s	1m37.006s
insig2	1m45.422s	0m01.561s	0m01.585s
irf1	0m22.771s	0m22.654s	0m22.830s
itfg3	1m47.671s	1m45.937s	1m38.529s
kcnk4	0m22.375s	0m22.189s	0m22.264s
leap2	0m00.741s	0m00.744s	0m00.736s
lep	0m53.622s	0m52.904s	0m53.748s
lif	0m11.987s	0m11.829s	0m11.815s
lrrc4	0m29.250s	0m29.435s	0m29.475s
lyzl6	0m16.846s	0m17.109s	0m16.394s
mdfi	1m13.691s	1m14.132s	1m15.353s
mettl2b	0m26.646s	0m26.209s	0m26.710s
mier3	3m59.459s	3m58.975s	4m01.091s
mmp26	0m02.732s	0m02.775s	0m02.758s
mrpl23	0m54.183s	0m54.226s	0m53.530s
mzf1	0m10.560s	0m10.623s	0m10.625s
nipal1	1m48.931s	1m51.296s	1m48.590s
nme4	0m06.398s	0m06.356s	0m06.225s
nr2e1	1m26.218s	1m23.158s	1m28.575s
nsdhl	1m36.586s	1m37.465s	1m36.868s
ooep	0m00.739s	0m00.759s	0m00.741s
or51g2	0m00.463s	0m00.479s	0m00.465s
or51m1	0m00.543s	0m00.549s	0m00.539s
or51q1	0m00.472s	0m00.474s	0m00.471s
or51s1	0m00.465s	0m00.472s	0m00.463s
or51t1	0m00.612s	0m00.632s	0m00.622s
or52h1	0m00.469s	0m00.479s	0m00.457s
or52j3	0m00.503s	0m00.507s	0m00.513s
osm	0m02.854s	0m02.889s	0m02.822s
ostm1	2m29.802s	2m32.207s	2m23.210s
pdia2	0m04.616s	0m04.549s	0m04.653s
pdk4	1m18.875s	1m16.670s	1m17.466s
pdx1	0m15.330s	0m15.440s	0m15.548s
pes1	0m49.494s	0m49.144s	0m49.051s
pex12	0m07.128s	0m07.158s	0m07.339s
pla2g3	0m07.621s	0m07.603s	0m07.690s
pnma3	0m01.601s	0m01.622s	0m01.593s
pnma5	0m01.963s	0m02.045s	0m01.955s
polr3k	0m07.871s	0m07.870s	0m07.838s

continua na próxima página

Gene	Boruvka	Kruskal	Prim
pon1	3m45.099s	3m46.406s	4m06.512s
pon3	5m25.767s	5m00.490s	5m21.788s
ppp1r14b	0m04.153s	0m04.118s	0m04.225s
ppp1r3a	6m54.948s	6m54.616s	7m38.387s
prickle4	0m03.741s	0m03.780s	0m03.731s
rasl10b	0m16.629s	0m16.744s	0m17.390s
rbm28	6m25.644s	6m19.309s	6m44.367s
rps5	0m11.652s	0m11.860s	0m11.935s
samd9	0m41.256s	0m40.695s	0m27.357s
samd9l	0m28.246s	0m29.134s	0m40.638s
sec14l3	0m39.198s	0m39.393s	0m39.368s
sec14l4	0m32.592s	0m32.343s	0m32.182s
selm	0m04.480s	0m04.512s	0m04.433s
shroom1	0m13.147s	0m12.958s	0m13.398s
slc22a11	0m10.643s	0m10.978s	0m10.920s
slc22a5	1m32.423s	1m39.230s	1m39.815s
slc27a5	0m05.699s	0m05.773s	0m05.664s
slc35e4	0m21.534s	0m21.276s	0m20.973s
slc4a3	1m02.119s	1m00.501s	1m00.151s
slfn13	0m10.210s	0m10.748s	0m10.592s
slfn14	0m15.660s	0m15.380s	0m15.659s
snrnp25	0m09.550s	0m09.210s	0m09.384s
snx19	7m10.834s	7m12.212s	7m03.377s
spp2	1m25.797s	1m25.594s	1m25.048s
steap1	0m47.159s	0m47.201s	0m46.908s
stip1	1m03.260s	1m01.908s	1m02.199s
syt8	0m03.479s	0m03.394s	0m03.519s
taf15	0m40.372s	0m39.940s	0m40.340s
tbc1d10a	3m43.275s	4m03.958s	3m41.809s
tcn2	0m43.977s	0m43.448s	0m46.175s
tfpi2	0m25.979s	0m25.820s	0m26.288s
tnni2	0m03.217s	0m03.226s	0m03.214s
tomm6	0m04.828s	0m04.863s	0m04.876s
trex2	0m00.827s	0m00.867s	0m00.822s
trim28	0m02.318s	0m02.332s	0m02.327s
ube2m	0m03.477s	0m03.673s	0m03.498s
ubqln3	0m01.810s	0m01.790s	0m01.782s
ubqlnl	0m01.727s	0m01.722s	0m01.724s
vegfb	0m01.717s	0m01.737s	0m01.740s
zbtb45	0m04.862s	0m04.934s	0m04.929s
zfp92	0m09.875s	0m10.237s	0m10.658s
zmat5	1m43.653s	1m47.316s	1m42.949s
znf132	0m06.669s	0m06.715s	0m06.643s
znf135	0m05.826s	0m05.882s	0m05.826s
znf324	0m05.516s	0m05.693s	0m05.515s
znf418	0m03.487s	0m03.681s	0m03.534s
znf446	0m04.745s	0m04.780s	0m04.744s
znf584	0m18.092s	0m16.958s	0m17.413s
znf622	0m45.296s	0m46.071s	0m45.638s
znf814	0m14.033s	0m13.835s	0m14.748s
znf8	0m04.530s	0m04.551s	0m04.509s
zscan22	0m14.238s	0m14.315s	0m14.122s

continua na próxima página

Gene	Boruvka	Kruskal	Prim
zscan4	0m02.448s	0m02.456s	0m02.457s

Tabela C.1: Listagem dos tempos de execução da ferramenta desenvolvida, para cada gene do conjunto de testes, utilizando os algoritmos de Boruvka, Kruskal e Prim. Na coluna Gene são listados os nomes dos genes. Na próximas três colunas, o tempo de execução que a ferramenta levou para gerar o cladograma, de acordo com os algoritmos de Boruvka, Kruskal e Prim, respectivamente.

Anexo D Códigos Fonte

Arquivo Fonte 1: Alinhamento.java

```
/**
 * Classe Alinhamento, realiza o alinhamento duas a duas sequencias e grava em
 * uma matriz de inteiro.
 *
 *
 * @author Helder Donizete da Silva Carvalho
 * @author Marcelo Martins
 */
public class Alinhamento {

    private int[][] matrizCusto;

    /**
     * Construtor para montar a matriz de Custos
     *
     * @param align parametro para informar qual metodo de alinhamento sera
     * usado.
     * @param sequencias vetor com as sequencias para ser calculada os custos.
     * @param gap valor para o <i>GAP</i>
     * @param match valor para o <i>MATCH</i>
     * @param mismatch valor para o <i>MISMATCH</i>
     */
    public Alinhamento(String align, Sequencia[] sequencias, int gap, int match, int
        mismatch) {
        int n = sequencias.length;
        int[][] temp;
        this.matrizCusto = new int[n][n];
        switch (align) {
            case "AlignGlobal":
                for (int i = 0; i < n; i++) {
                    for (int j = 0; j < n; j++) {
                        if (i == j) {
                            this.matrizCusto[i][j] = Integer.MAX_VALUE;
                        } else {
                            temp = AlignGlobal(sequencias[i].getSequencia(), sequencias[
                                j].getSequencia(), gap, match, mismatch);
                            this.matrizCusto[i][j] = temp[sequencias[i].getTamanho()][
                                sequencias[j].getTamanho()] * -1;
                        }
                    }
                }
                break;
            case "AlignLocal":
                for (int i = 0; i < n; i++) {
                    for (int j = 0; j < n; j++) {
                        if (i == j) {
                            this.matrizCusto[i][j] = Integer.MAX_VALUE;
                        } else {
                            temp = AlignLocal(sequencias[i].getSequencia(), sequencias[j
                                ].getSequencia(), gap, match, mismatch);
                            this.matrizCusto[i][j] = temp[sequencias[i].getTamanho()][
                                sequencias[j].getTamanho()] * -1;
                        }
                    }
                }
                break;
        }
    }

    /**
     * Metodo de alinhamento global de programacao dinamica de Saul Needleman e
     * Christian Wunsch
     *
     * @param sequencia1 Sequencia 01 para fazer o calculo do alinhamento
     * @param sequencia2 Sequencia 02 para fazer o calculo do alinhamento
     * @param gap valor do gap
```

```
     * @param match valor do match
     * @param mismatch valor do mismatch
     * @return a Matriz de Distancias do objeto
     */
    private int[][] AlignGlobal(StringBuilder sequencia1, StringBuilder sequencia2, int
        gap, int match, int mismatch) {
        int[][] matriz;
        int n = sequencia1.length();
        int m = sequencia2.length();
        int i, j, max;

        matriz = new int[n + 1][m + 1];
        for (i = 0; i < n + 1; i++) {
            matriz[i][0] = gap * i;
        }
        for (i = 1; i < m + 1; i++) {
            matriz[0][i] = gap * i;
        }
        for (i = 1; i < n + 1; i++) {
            for (j = 1; j < m + 1; j++) {
                max = matriz[i][j - 1] + gap;
                if (sequencia1.charAt(i - 1) == sequencia2.charAt(j - 1)) {
                    if (max < matriz[i - 1][j - 1] + match) {
                        max = matriz[i - 1][j - 1] + match;
                    }
                } else {
                    if (max < matriz[i - 1][j - 1] + mismatch) {
                        max = matriz[i - 1][j - 1] + mismatch;
                    }
                }
                if (max < matriz[i - 1][j] + gap) {
                    max = matriz[i - 1][j] + gap;
                }
                matriz[i][j] = max;

            }
        }
        return matriz;
    }

    /**
     * Metodo de alinhamento de programacao dinamica Local
     *
     * @param sequencia1 Sequencia 01 para fazer o calculo do alinhamento
     * @param sequencia2 Sequencia 02 para fazer o calculo do alinhamento
     * @param gap valor do gap
     * @param match valor do match
     * @param mismatch valor do mismatch
     * @return a Matriz de Distancias do objeto
     */
    private int[][] AlignLocal(StringBuilder sequencia1, StringBuilder sequencia2, int
        gap, int match, int mismatch) {
        int[][] matriz;
        int n = sequencia1.length();
        int m = sequencia2.length();
        int i, j, max;

        matriz = new int[n + 1][m + 1];
        for (i = 0; i < n + 1; i++) {
            matriz[i][0] = 0;
        }
        for (i = 1; i < m + 1; i++) {
            matriz[0][i] = 0;
        }
        for (i = 1; i < n + 1; i++) {
            for (j = 1; j < m + 1; j++) {
                max = matriz[i][j - 1] + gap;
                if (sequencia1.charAt(i - 1) == sequencia2.charAt(j - 1)) {
                    if (max < matriz[i - 1][j - 1] + match) {
                        max = matriz[i - 1][j - 1] + match;
```

```
                    }
                } else {
                    if (max < matriz[i - 1][j - 1] + mismatch) {
                        max = matriz[i - 1][j - 1] + mismatch;
                    }
                }
                if (max < matriz[i - 1][j] + gap) {
                    max = matriz[i - 1][j] + gap;
                }
                matriz[i][j] = max;
            }
        }
        return matriz;
    }

    public int[][] getMatrizCusto() {
        return this.matrizCusto;
    }
}
```

Arquivo Fonte 2: Aresta.java

```
/**
 * Classe aresta, objeto para gravar as informacoes das arestas
 *
 *
 * @author Helder Donizete da Silva Carvalho
 * @author Marcelo Martins
 */
public class Aresta {

    private int v1;
    private int v2;
    private int custo;

    /**
     * Construtor da classe aresta para gerar o objeto com as informacoes
     *
     * @param v1 parametro que sera gravado como vertice 1
     * @param v2 parametro que sera gravado como vertice 2
     * @param custo parametro que ser? gravado como o custo da aresta
     */
    public Aresta(int v1, int v2, int custo) {
        this.v1 = v1;
        this.v2 = v2;
        this.custo = custo;
    }

    public int getV1() {
        return v1;
    }

    public void setV1(int v1) {
        this.v1 = v1;
    }

    public int getV2() {
        return v2;
    }

    public void setV2(int v2) {
        this.v2 = v2;
    }

    public int getCusto() {
        return custo;
    }

    public void setCusto(int custo) {
        this.custo = custo;
```

```
    }
}
```

Arquivo Fonte 3: ArvFilo.java

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.util.LinkedList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;

/**
 * Classe statica ArvFilo, que e a classe principal da ferramenta, onde o codigo
 * esta organizado para realizar as chamadas como descrito no trabalho, para que
 * seja criada uma arvore filogenetica.
 *
 * @author Helder Donizete da Silva Carvalho
 * @author Marcelo Martins
 */
public class ArvFilo {

    /**
     * Metodo estatico main, para a geracao da arvore filogenetica.
     *
     * @param args sera responsavel pela entrado sistema. Os parametros serao
     * serapados por espaco, sendo o primeiro item sera a quantidades de
     * amostras, na sequencia as amostras utilizadas.<br>
     * Depois ira informar qual MST devera ser gerada (<i>Boruvka, Kruskal</i>
     * ou <i>Prim</i>).<br>
     * Em seguida os tres proximos valores informados serao os valores do
     * <i>GAP, MATCH e MISMATCH</i>.
     */
    @SuppressWarnings({"empty-statement", "UnusedAssignment"})
    public static void main(String[] args) {
        int qtdSeq, gap, match, mismatch;
        Sequencia[] vetorSeq;
        int matrizCusto[][];
        String alinhamentoUsado = "AlignGlobal";

        if (args.length > 0) {

            /*
             Ordem dos parametros
             args[0] = quantidade de sequencias que serao analisadas (definindo como "n
                 ")
             args[1] ate args[n] = nome dos arquivos que contem as sequencias
             args[n+1] = gap
             args[n+2] = match
             args[n+3] = mismatch
             args[n+4] = MST
             args[n+5] = nome do gene utilizado
             */
            qtdSeq = Integer.parseInt(args[0]);
            vetorSeq = new Sequencia[qtdSeq];
            matrizCusto = new int[qtdSeq][qtdSeq];

            //criando o vetor com as sequencias
            for (int i = 0; i < qtdSeq; i++) {
                vetorSeq[i] = new Sequencia(i, args[i + 1]);
            }

            /**
             * Definindo os valores de <i> GAP, MATCH </i> e <i>MISMATCH</i>
             * nesse sequencia.
             */
            gap = Integer.parseInt(args[qtdSeq + 1]);
            match = Integer.parseInt(args[qtdSeq + 2]);
```

```
65              mismatch = Integer.parseInt(args[qtdSeq + 3]);

                /**
                 * Se faz necessario definir quais tipos de alinhamento serao
                 * utilizados
                 *
                 * Ha dois construtores na Classe Alinhamento Primeiro - informar o
                 * nome do alinhamento a ser utilizado e o vetor de Sequencias
                 * Segundo - informar somente o vetor de alinhamento
                 */
                long start, end;
                start = System.currentTimeMillis();
                Alinhamento alinhamento = new Alinhamento(alinhamentoUsado, vetorSeq, gap,
                    match, mismatch);
                matrizCusto = alinhamento.getMatrizCusto();

                int minimo = Integer.MAX_VALUE;
                for (int i = 0; i < matrizCusto.length; i++) {
                    for (int j = 0; j < matrizCusto.length; j++) {
                        if (minimo > matrizCusto[i][j]) {
                            minimo = matrizCusto[i][j];
                        }
                    }
                }

                //organiza a matrizCusto antes de gerar o arquivo para o R e cria o arquivo
                    .csv para o R
                int matrizCustoR[][] = matrizCusto;

                gerarMatrizR(matrizCusto, matrizCustoR, "Align", minimo, "matriz", vetorSeq)
                    ;

                MST tree;
                String nome = null;
                switch (args[qtdSeq + 4]) {
                    case "Boruvka":
                        tree = new Boruvka(matrizCusto);
                        System.out.println("Boruvka x " + args[qtdSeq + 5]);
                        nome = "Boruvka";
                        break;
                    case "Kruskal":
                        tree = new Kruskal(matrizCusto);
                        System.out.println("Kruskal x " + args[qtdSeq + 5]);
                        nome = "Kruskal";
                        break;
                    case "Prim":
                        tree = new Prim(matrizCusto, vetorSeq);
                        System.out.println("Prim x " + args[qtdSeq + 5]);
                        nome = "Prim";
                        break;
                    default:
                        tree = null;
                        break;
                }

                matrizCusto = null;
                matrizCustoR = null;
                System.gc();

                //criar arquivo da matriz com o resultado da MST matrizMST.csv
                minimo = Integer.MAX_VALUE;
                for (Aresta a : tree.getMst()) {
                    if (minimo > a.getCusto()) {
                        minimo = a.getCusto();
                    }
                }

                gerarMatrizMST(tree.getMst(), minimo, "MST", "matriz", vetorSeq);

                //Iniciando a operacao pelo R.
```

```
            Runtime run = Runtime.getRuntime();
            try {
                Process p
                        = run.exec("/usr/lib/R/bin/Rscript --vanilla script.r");
                p.waitFor();

            } catch (Exception ex) {
                Logger.getLogger(ArvFilo.class.getName()).log(Level.SEVERE, null, ex);
                System.err.println("ERRO");
            }

            //Executando script para gerar as correlacoes
            try {
                Process p
                        = run.exec("/usr/lib/R/bin/Rscript --vanilla script2.r");
                p.waitFor();

            } catch (Exception ex) {
                Logger.getLogger(ArvFilo.class.getName()).log(Level.SEVERE, null, ex);
                System.err.println("ERRO");
            }

            end = System.currentTimeMillis();

            //exibindo os valores
            LinkedList<Aresta> resultado = tree.getMst();
            for (Aresta a : resultado) {
                System.out.println(a.getV1() + " " + a.getV2() + " " + a.getCusto());
            }
            System.out.println("Tempo: " + (end - start));
            System.out.println("");

            ImageIcon img = new ImageIcon("dendrograma.png");
            if (img.getIconHeight() != -1 && img.getIconWidth() != -1) {
                JFrame tela = new JFrame("Cladograma: " + nome);
                JLabel dendro = new JLabel();
                tela.setSize(img.getIconHeight(), img.getIconWidth());
                tela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                tela.setAlwaysOnTop(true);
                tela.setLocationRelativeTo(null);
                dendro.setIcon(img);
                tela.getContentPane().add(dendro);
                tela.setVisible(true);
            }

            System.gc();

        } else {
            System.out.println("COMANDO INVALIDO");

        }
    }

    /**
     * Metodo para criar o arquivo da matriz para ser usado pelo R para analise
     *
     * @param matrizCusto matriz para ser analisada
     * @param tipo identificador para verificar a origem dessa matriz (de
     * Alinhamento ou da MST)
     * @param gene qual o gente que esta sendo analisado
     * @param vetorSeq vetor de sequencias do grafo
     */
    private static void gerarArquivo(int[][] matrizCusto, String tipo, String gene,
        Sequencia[] vetorSeq) {

        String file = gene + tipo + ".csv";
        StringBuilder line = new StringBuilder();
        int i, j;

        try {
```

```
            try (BufferedWriter buff = new BufferedWriter(new FileWriter(file))) {
                for (i = 0; i < vetorSeq.length; i++) {
                    line = line.append("\"");
                    line = line.append(vetorSeq[i].getNome());
                    line = line.append("\"");
                    if (i != matrizCusto.length - 1) {
                        line = line.append(",");
                    }
                }
                buff.write(line.toString());
                buff.newLine();
                line.delete(0, line.length());
                for (i = 0; i < matrizCusto.length; i++) {
                    for (j = 0; j < matrizCusto.length; j++) {
                        line = line.append(matrizCusto[i][j]);
                        if (j != matrizCusto.length - 1) {
                            line = line.append(",");
                        }
                    }
                    buff.write(line.toString());
                    buff.newLine();
                    line.delete(0, line.length());
                }
            }
        } catch (Exception e) {
        }
    }

    /**
     * Metodo para gerar a Matriz de acordo que o R necessita
     *
     * @param matrizCusto Matriz que esta sendo trabalhada pelo programa para
     * ser convertida
     * @param matrizCustoR Matriz que sera para o R
     * @param align nome do alinhamento
     * @param minimo valor minimo da matrizCusto
     * @param gene nome do gene analisado
     * @param vetorSeq vetor de sequencias do grafo
     */
    private static void gerarMatrizR(int[][] matrizCusto, int[][] matrizCustoR, String
        align, int minimo, String gene, Sequencia[] vetorSeq) {
        minimo = minimo - 50;

        for (int i = 0; i < matrizCusto.length; i++) {
            for (int j = 0; j < matrizCusto.length; j++) {
                if (matrizCusto[i][j] == Integer.MAX_VALUE) {
                    matrizCustoR[i][j] = matrizCusto[i][j];
                } else {
                    matrizCustoR[i][j] = matrizCusto[i][j] - minimo;
                }
            }
        }

        gerarArquivo(matrizCustoR, align, gene, vetorSeq);
    }

    /**
     * Metodo para transformar as informacoes da MST em uma matriz
     *
     * @param rk lista de arestas da MST
     * @param minimo valor minimo de custo da MST
     * @param tipo tipo do Arquivo para o nome
     * @param gene nome do gene
     * @param vetorSeq vetor de sequencias do grafo
     */
    private static void gerarMatrizMST(LinkedList<Aresta> rk, int minimo, String tipo,
        String gene, Sequencia[] vetorSeq) {
        int matriz[][] = new int[vetorSeq.length][vetorSeq.length + 1];
        minimo = minimo - 50;
```

```
270             for (int i = 0; i < matriz.length; i++) {
                    for (int j = 0; j < matriz.length; j++) {
                        matriz[i][j] = Integer.MAX_VALUE;
                    }
                }
275
                for (Aresta a : rk) {
                    matriz[a.getV1()][a.getV2()] = a.getCusto() - minimo;
                    matriz[a.getV2()][a.getV1()] = a.getCusto() - minimo;
                }
280
                gerarArquivo(matriz, tipo, gene, vetorSeq);
            }
        }
```

Arquivo Fonte 4: Boruvka.java

```
1   import java.util.Iterator;
    import java.util.LinkedList;
    import java.util.logging.Level;
    import java.util.logging.Logger;
5
    /**
     * Classe Boruvka, classe que herda os metodos da classe MST. Classe Boruvka, e
     * a classe que gera uma MST baseado no alg. de Boruvka
     *
10   * @author Helder Donizete da Silva Carvalho
     * @author Marcelo Martins
     */
    public class Boruvka extends MST {

15      private LinkedList<Aresta> mst = new LinkedList<>();

        /**
         * Construtor que gerar a MST baseado no algoritmo de Boruvka
         *
20       * @param matriz matriz de Custo gerado entre as sequencias
         */
        @SuppressWarnings({"rawtype","unchecked"})
        Boruvka(int[][] matriz) {
            LinkedList<LinkedList> S = new LinkedList<>();
25          LinkedList<Integer> vertex;
            LinkedList<Aresta> arestas;
            Aresta aux;

            vertex = gerarListVertice(matriz);
30
            for (int v : vertex) {
                try {
                    makeSet(v, S);
                } catch (Exception ex) {
35                  Logger.getLogger(Boruvka.class.getName()).log(Level.SEVERE, null, ex);
                }
            }

            arestas = gerarArestas(matriz);
40          Iterator it = S.iterator();

            while (mst.size() < vertex.size() - 1) {
                aux = extractMin( (LinkedList) it.next(), vertex, arestas);
                if (findTheSet(aux.getV1(), S) != findTheSet(aux.getV2(), S)) {
45                  mst.add(aux);
                    union(aux.getV1(), aux.getV2(), S);
                    arestas.remove(aux);
                }
                it = S.iterator();
50          }
        }
```

```
    @Override
    @SuppressWarnings({"rawtype","unchecked"})
    public LinkedList<Aresta> getMst() {
        return mst;
    }

}
```

Arquivo Fonte 5: Kruskal.java

```
import java.util.LinkedList;
import java.util.PriorityQueue;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * Classe Kruskal, classe que herda os metodos da classe MST. Classe Kruskal, e
 * a classe que gera uma MST baseado no alg. de Kruskal
 *
 * @author Helder Donizete da Silva Carvalho
 * @author Marcelo Martins
 */
public class Kruskal extends MST {

    private LinkedList<Aresta> mst = new LinkedList<>();

    /**
     * Construtor que gerar a MST baseado no algoritmo de Kruskal
     *
     * @param matriz matriz de Custo gerado entre as sequencias
     */
    @SuppressWarnings("unchecked")
    public Kruskal(int[][] matriz) {
        LinkedList<LinkedList> S = new LinkedList<>();
        PriorityQueue<Integer> edges;
        LinkedList<Integer> vertex;
        LinkedList<Aresta> arestas;
        Aresta aux;

        vertex = gerarListVertice(matriz);

        for (int v : vertex) {
            try {
                makeSet(v, S);
            } catch (Exception ex) {
                Logger.getLogger(Kruskal.class.getName()).log(Level.SEVERE, null, ex);
            }
        }

        edges = heapArestas(matriz);
        arestas = gerarArestas(matriz);

        aux = retornaAresta(edges.poll(), arestas);
        mst.add(aux);
        union(aux.getV1(), aux.getV2(), S);
        arestas.remove(aux);

        while (edges.size() > 0) {
            aux = retornaAresta(edges.poll(), arestas);
            if (findTheSet(aux.getV1(), S) != findTheSet(aux.getV2(), S)) {
                mst.add(aux);
                union(aux.getV1(), aux.getV2(), S);

            }
            arestas.remove(aux);
        }
    }

    @Override
    @SuppressWarnings({"rawtype","unchecked"})
```

```
    public LinkedList<Aresta> getMst() {
        return mst;
    }

}
```

Arquivo Fonte 6: MST.java

```
import java.util.Collections;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.PriorityQueue;

/**
 * Classe abstrata MST, que contem os metodos necessarios
 * para a implementacao algoritmos de MST
 *
 * @author Helder Donizete da Silva Carvalho
 * @author Marcelo Martins
 */

abstract class MST {

    LinkedList<Aresta> getMst(){
        return null;
    }

    /**
     * Metodo que a partir do custo retorna a aresta correspondente.
     *
     * @param custo custo da aresta.
     * @param lista lista de arestas.
     * @return a aresta encontrada.
     */
    @SuppressWarnings({"rawtype","unchecked"})
    public Aresta retornaAresta(int custo, LinkedList<Aresta> lista) {
        Aresta aresta = null;

        for (Aresta aux : lista) {
            if (aux.getCusto() == custo) {
                return aresta = aux;
            }
        }

        return null;
    }

    /**
     * Metodo que a partir da matriz gera a lista de vertices(sequencias) do
     * grafo. <br>
     * No caso a lista tera somente os indices dos vertices.
     *
     * @param m Matriz de adjacencias
     * @return Lista de Vertices do grafo
     */
    @SuppressWarnings({"rawtype","unchecked"})
    public LinkedList<Integer> gerarListVertice(int m[][]) {
        LinkedList<Integer> vertex = new LinkedList<Integer>();
        int n = m[0].length;

        for (int i = 0; i < m[0].length; i++) {
            vertex.add(i);
        }

        return vertex;
    }

    /**
     * Metodo que a partir da matriz obtem a lista de arestas ordenadas de forma
     * nao decrescente
```

```
     *
     * @param m Matriz de adjacencias
     * @param sequencias lista de sequencias
     * @return Lista de Arestas ordenadas
     */
    @SuppressWarnings({"rawtype","unchecked"})
    public LinkedList<Integer> ordenarArestas(int[][] m, Sequencia[] sequencias) {
        LinkedList<Integer> edge = new LinkedList<Integer>();
        int qtdV = m[0].length;

        for (int i = 0; i < qtdV - 1; i++) {
            for (int j = 1; j < qtdV; j++) {
                edge.add(m[i][j]);
            }
        }

        Collections.sort(edge);
        return edge;
    }

    /**
     * Metodo que a partir da matriz obtem um heap dos custos de arestas
     *
     * @param m Matriz de adjacencias
     * @return Heap dos custos das arestas
     */
    @SuppressWarnings({"rawtype","unchecked"})
    public PriorityQueue<Integer> heapArestas(int[][] m) {
        PriorityQueue<Integer> edge = new PriorityQueue<>();
        int qtdV = m[0].length;

        for (int i = 0; i < qtdV - 1; i++) {
            for (int j = 1; j < qtdV; j++) {
                edge.offer(m[i][j]);
            }
        }

        return edge;
    }

    /**
     * Metodo que a partir da matriz de custo e do vetor de sequencias obtem a
     * lista de arestas
     *
     * @param m Matriz de Adjacencias
     * @return retorna a lista de objetos aresta.
     */
    @SuppressWarnings({"rawtype","unchecked"})
    public LinkedList<Aresta> gerarArestas(int[][] m) {
        LinkedList<Aresta> arestas = new LinkedList<>();
        Aresta aresta;
        int qtdV = m[0].length;

        for (int i = 0; i < qtdV - 1; i++) {
            for (int j = 1; j < qtdV; j++) {
                arestas.add(aresta = new Aresta(i, j, m[i][j]));
            }
        }

        return arestas;
    }

    /**
     * Extrai a aresta adjacente ao Conjunto com menor custo
     *
     * @param S Conjunto de vertices da mst que esta sendo formado
     * @param vertices Conjunto de vertices para ser adicionado
     * @param arestas lista de arestas do grafo
     * @return A aresta de menor custo que sera adicionado a MST
     */
```

```
    @SuppressWarnings({"rawtype","unchecked"})
    public Aresta extractMin(LinkedList<Integer> S, LinkedList<Integer> vertices,
        LinkedList<Aresta> arestas) {
135     //inicializar com custo maximo para o primeiro caso
        Aresta u = new Aresta(0, 0, Integer.MAX_VALUE);
        int temp, temp2; // guarda o vertice atual a ser verificado
        boolean teste = false;
        boolean laco = true;
140     LinkedList<Integer> mst = (LinkedList) S.clone();

        while (mst.size() > 0) {
            LinkedList<Integer> vadj = adj( mst.getFirst(), arestas, S);

145         for (int i : vadj) {
                for (Aresta e2 : arestas) {
                    if (i == e2.getV1()) {
                        for (int i2 : S) {
                            if (i2 == e2.getV2()) {
150                             teste = true;
                            }
                        }
                    } else if (i == e2.getV2()) {
                        for (int i3 : S) {
155                         if (i3 == e2.getV1()) {
                                teste = true;
                            }
                        }
                    }
160                 if (teste) {
                        if (u.getCusto() > e2.getCusto()) {
                            u.setV1(e2.getV1());
                            u.setV2(e2.getV2());
                            u.setCusto(e2.getCusto());
165                     }
                    }
                    teste = false;
                }
            }
170         mst.removeFirst();
        }
        return u;
    }

175

    /**
     * Metodo que a partir de um vertice v, busca quais as arestas sao adjacentes
     * sem que estejam no conjunto S
     *
180  * @param v Vertice de origem para verificar adjacentes
     * @param arestas lista de arestas do grafo
     * @param S Conjunto de Vertices que estao no conjunto resposta
     */
    @SuppressWarnings({"rawtype","unchecked"})
185 public LinkedList<Integer> adj(int v, LinkedList<Aresta> arestas, LinkedList<Integer
        > S) {
        LinkedList<Integer> list = new LinkedList<Integer>();

        for (Aresta e : arestas) {
            if (e.getV1() == v) {
190

                if (!S.contains(e.getV2())) {
                    list.add(e.getV2());
                }

195         } else if (e.getV2() == v) {
                if (!S.contains(e.getV1())) {
                    list.add(e.getV1());
                }

200         }
```

```
            }
            return list;

        }

        /**
         * Metodo que a partir de dois Vertices x e y verificam quais
         * sao seus respectivos conjuntos e ai fazem a uniao dos mesmos
         *
         * @param x Vertice 1
         * @param y Vertice 2
         * @param sets Conjunto de Conjuntos
         */
        @SuppressWarnings({"rawtype","unchecked"})
        public void union(int x, int y, LinkedList<LinkedList> sets) {

            // Procura os conjuntos de x e y
            LinkedList<Integer> sx = findTheSet(x, sets);
            LinkedList<Integer> sy = findTheSet(y, sets);

            // Adiciona os membros da lista de x na lista xy
            LinkedList<Integer> sxy = new LinkedList<Integer>(sx);
            // Adiciona os membros da lista de y na lista xy
            for (int yy : sy) {
                sxy.add(yy);
            }

            sets.add(sxy);

            // Remove Sx e Sy
            sets.remove(sx);
            sets.remove(sy);
        }

        /**
         * Metodo que transforma cada elemente x, em conjunto em sets
         *
         * @param x Vertice
         * @param sets Conjunto em que x vai ser adicionado
         */
        @SuppressWarnings({"rawtype","unchecked"})
        public void makeSet(int x, LinkedList<LinkedList> sets) throws Exception {

            if (findSet(x, sets) != null) {
                throw new Exception("MakeSet: elemento ja criado pelo makeSet!");
            }

            LinkedList<Integer> si = new LinkedList<Integer>();
            si.add(x);
            sets.add(si);
        }

        /**
         * Metodo que procura no conjunto de conjuntos sets
         * se o elemente x esta em algum conjunto de sets
         *
         * @param x Vertice para ser localizado
         * @param sets Conjunto de conjuntos
         * @return Retorna o vertice que esta sendo procurado
         */
        @SuppressWarnings("rawtype")
        public Object findSet(Integer x, LinkedList<LinkedList> sets) {
            LinkedList<Integer> s = findTheSet(x, sets);

            if (s == null) {
                return null;
            }

            Iterator<Integer> i = s.iterator();
            if (i.hasNext()) {
```

```
            return i.next();
        } else {
            return null;
        }
    }

    /**
     * Metodo que retorna o Conjunto que esta o Vertice x
     *
     * @param x Vertice para ser procurado
     * @param sets Conjunto de conjuntos
     * @return Retorna o conjunto do Vertice que esta sendo procurado
     */
    @SuppressWarnings({"rawtype","unchecked"})
    LinkedList<Integer> findTheSet(int x, LinkedList<LinkedList> sets) {
        Iterator i = sets.iterator();
        LinkedList<Integer> s = null;
        while (i.hasNext()) {
            LinkedList<Integer>                    si = (LinkedList) i.next();
            if (si.contains(x)) {
                s = si;
                break;
            }
        }

        return s;
    }
}
```

Arquivo Fonte 7: Prim.java

```
import java.util.LinkedList;
import java.util.PriorityQueue;

/**
 * Classe Prim, classe que herda os metodos da classe MST. Classe Prim, e a
 * classe que gera uma MST baseado no alg. de Prim
 *
 * @author Helder Donizete da Silva Carvalho
 * @author Marcelo Martins
 */
public class Prim extends MST {

    private LinkedList<Aresta> mst = new LinkedList<>();

    /**
     * Construtor que gerar a MST baseado no algoritmo de Prim
     *
     * @param sequencias vetor de sequencias
     * @param m matriz de Custo gerado entre as sequencias
     */
    @SuppressWarnings({"rawtype","unchecked"})
    public Prim(int[][] m, Sequencia[] sequencias) {
        LinkedList<Integer> vertex;
        LinkedList<Integer> mstP = new LinkedList<>();
        LinkedList<Aresta> arestas;
        PriorityQueue<Integer> edges = new PriorityQueue<>();
        Aresta aux;

        vertex = gerarListVertice(m);
        edges = heapArestas(m);
        arestas = gerarArestas(m);

        int a = vertex.remove();
        mstP.add(a);

        while (mst.size() < vertex.size()) {
            aux = extractMin(mstP, vertex, arestas);
            mst.add(aux);
```

```
            if (mstP.contains(aux.getV1())) {
                mstP.add(aux.getV2());
            } else {
                mstP.add(aux.getV1());
            }
        }
    }

    @SuppressWarnings({"rawtype","unchecked"})
    public LinkedList<Aresta> getMst() {
        return mst;
    }

}
```

Arquivo Fonte 8: Sequencia.java

```
import java.io.BufferedReader;
import java.io.FileReader;

/**
 * Classe Sequencia, objeto que esta gravada as informacoes de cada sequencia
 *
 * @author Helder Donizete da Silva Carvalho
 * @author Marcelo Martins
 */
public class Sequencia {

    private String nome;
    private StringBuilder sequencia = new StringBuilder();
    private int tamanho;
    private int indice;

    /**
     * Construtor para gerar as informacoes da sequencia
     *
     * @param i posicao do vetor;
     * @param arquivo nome do arquivo para ser extraido as informacoes da
     * sequencia.
     */
    public Sequencia(int i, String arquivo) {
        this.indice = i;
        String aux1[];
        aux1 = arquivo.split("/|_|\\.");

        try {

            FileReader file = new FileReader(arquivo);
            BufferedReader buff = new BufferedReader(file);
            String line = buff.readLine();
            String aux[];

            while (!line.equals("")) {
                if (line.trim().length() != 0) {
                    this.sequencia = this.sequencia.append(line);
                }
                line = buff.readLine();
            }
            buff.close();
        } catch (Exception e) {
        }
        this.tamanho = sequencia.length();
        this.nome = aux1[aux1.length - 2];
    }

    public String getNome() {
        return nome;
    }

    public StringBuilder getSequencia() {
```

```
            return sequencia;
        }

        public int getTamanho() {
            return tamanho;
        }

        public int getIndice() {
            return indice;
        }
    }
```